Docket No. 240490US2/hyc

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Tsutomu OHISHI, et al. GAU: 2622

SERIAL NO: 10/621,450 EXAMINER:

FILED: July 18, 2003

FOR: IMAGE FORMING APPARATUS, WRAPPING METHOD AND THE PROGRAM

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of **35 U.S.C. §119(e)**: **Application No.** **Date Filed**

■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
JAPAN	2002-211685	July 19, 2002
JAPAN	2003-196228	July 14, 2003

Certified copies of the corresponding Convention Application(s)

■ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Joseph A. Scafetta, Jr.

Marvin J. Spivak
Registration No. 24,913

Joseph A. Scafetta, Jr.
Registration No. 26,803

Customer Number
22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 05/03)

10/621,450

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００２年　７月１９日

出願番号
Application Number: 特願２００２－２１１６８５

[ST.10/C]: [ＪＰ２００２－２１１６８５]

出願人
Applicant(s): 株式会社リコー

２００３年　５月　６日

特許庁長官
Commissioner,
Japan Patent Office 太田信一郎

出証番号　出証特２００３－３０３２４３５

【書類名】　特許願
【整理番号】　0201298
【提出日】　平成14年 7月19日
【あて先】　特許庁長官殿
【国際特許分類】　G03G 21/00　370
【発明の名称】　画像形成装置およびラッピング処理方法並びにバージョン管理方法
【請求項の数】　33
【発明者】
【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
【氏名】　大石　勉
【発明者】
【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
【氏名】　秋吉　邦洋
【発明者】
【住所又は居所】　東京都大田区中馬込１丁目３番６号　株式会社リコー内
【氏名】　田中　浩行
【特許出願人】
【識別番号】　000006747
【氏名又は名称】　株式会社リコー
【代理人】
【識別番号】　100089118
【弁理士】
【氏名又は名称】　酒井　宏明
【手数料の表示】
【予納台帳番号】　036711
【納付金額】　21,000円
【提出物件の目録】
【物件名】　明細書　1

出証特２００３－３０３２４３５

【物件名】　　図面　1

【物件名】　　要約書　1

【包括委任状番号】　9808514

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　画像形成装置およびラッピング処理方法並びにバージョン管理方法

【特許請求の範囲】

【請求項１】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記コントロールサービスのいずれかのコンポーネントがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理手段

を備えたことを特徴とする画像形成装置。

【請求項２】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記アプリケーションプログラムインタフェースがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理手段

を備えたことを特徴とする画像形成装置。

【請求項３】　画像形成処理で使用されるハードウェア資源と、画像形成処

理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記コントロールサービスから前記アプリケーションに対するメッセージを取捨選択するラッピング処理手段

を備えたことを特徴とする画像形成装置。

【請求項４】　前記ラッピング処理手段は、関数または関数の引数が追加されてバージョンアップしたときに、追加された関数または関数の引数に対応する機能を無視して整合性をとることを特徴とする請求項１または２に記載の画像形成装置。

【請求項５】　前記ラッピング処理手段は、関数または関数の引数が分割されてバージョンアップしたときに、分割された関数または関数の引数に応じてダミーの関数または関数の引数を補填して整合性をとることを特徴とする請求項１または２に記載の画像形成装置。

【請求項６】　前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを予め設定可能にしたことを特徴とする請求項３に記載の画像形成装置。

【請求項７】　前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスをさらに備え、

前記ラッピング処理手段は、前記仮想アプリケーションサービスに含まれていることを特徴とする請求項１～６のいずれか一つに記載の画像形成装置。

【請求項８】　前記ラッピング処理手段は、前記仮想アプリケーションサービスのプロセス内部でスレッドとして生成されることを特徴とする請求項７に記載の画像形成装置。

【請求項９】　前記アプリケーションの実行時に、共通の関数を利用して他のプログラムと結合させて実行形式のプログラムを作成する動的リンク手段をさらに備えていることを特徴とする請求項１～８のいずれか一つに記載の画像形成装置。

【請求項１０】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記アプリケーションと前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段

を備えたことを特徴とする画像形成装置。

【請求項１１】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記アプリケーションと前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段

を備えたことを特徴とする画像形成装置。

【請求項１２】　前記バージョン管理手段は、サポート可能な範囲のバージョンか否かを前記アプリケーションのバージョン番号に基づいて判定することを特徴とする請求項１０または１１に記載の画像形成装置。

【請求項１３】　前記バージョン管理手段は、サポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルをさらに備え、前記アプリケーションのバージョン番号を前記バージョン管理テーブルと比較して判定することを特徴とする請求項１２に記載の画像形成装置。

【請求項１４】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段

を備えたことを特徴とする画像形成装置。

【請求項１５】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、

前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段

を備えたことを特徴とする画像形成装置。

【請求項１６】　前記バージョン管理手段は、サポート可能な範囲のバージョンか否かを前記アプリケーションが使用する全ての関数番号とバージョン番号とに基づいて判定することを特徴とする請求項１４または１５に記載の画像形成装置。

【請求項１７】　前記バージョン管理手段は、サポート可能な範囲の全ての関数番号とバージョン番号とが登録されたバージョン管理テーブルをさらに備え、前記アプリケーションが使用する全ての関数番号とバージョン番号を前記バージョン管理テーブルと比較して判定することを特徴とする請求項１６に記載の画像形成装置。

【請求項１８】　前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスをさらに備え、

前記バージョン管理手段は、前記仮想アプリケーションサービスに含まれていることを特徴とする請求項１０～１７のいずれか一つに記載の画像形成装置。

【請求項１９】　前記バージョン管理手段は、前記仮想アプリケーションサービスのプロセス内部でスレッドとして生成されることを特徴とする請求項１８に記載の画像形成装置。

【請求項２０】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とする

アプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとのバージョン差を吸収するラッピング処理方法であって、

前記コントロールサービスのいずれかのコンポーネントがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理ステップ

を含むことを特徴とするラッピング処理方法。

【請求項２１】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとのバージョン差を吸収するラッピング処理方法であって、

前記アプリケーションプログラムインタフェースがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理ステップ

を含むことを特徴とするラッピング処理方法。

【請求項２２】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとのバージョン差を吸収するラッピング処理方法であって、

前記コントロールサービスから前記アプリケーションに対するメッセージを取

捨選択するラッピング処理ステップ

を含むことを特徴とするラッピング処理方法。

【請求項２３】　前記ラッピング処理ステップは、関数または関数の引数が追加されてバージョンアップしたときに、追加された関数または関数の引数に対応する機能を無視して整合性をとることを特徴とする請求項２０または２１に記載のラッピング処理方法。

【請求項２４】　前記ラッピング処理ステップは、関数または関数の引数が分割されてバージョンアップしたときに、分割された関数または関数の引数に応じてダミーの関数または関数の引数を補填して整合性をとることを特徴とする請求項２０または２１に記載のラッピング処理方法。

【請求項２５】　前記ラッピング処理ステップは、前記アプリケーションに対して通知しない特定のメッセージを予め設定可能にしたことを特徴とする請求項２２に記載のラッピング処理方法。

【請求項２６】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとの間のバージョンを管理するバージョン管理方法であって、

前記アプリケーションと前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップ

を含むことを特徴とするバージョン管理方法。

【請求項２７】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプ

ロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとの間のバージョンを管理するバージョン管理方法であって、

前記アプリケーションと前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップ

を含むことを特徴とするバージョン管理方法。

【請求項２８】　前記バージョン管理ステップは、サポート可能な範囲のバージョン差か否かを前記アプリケーションのバージョン番号に基づいて判定することを特徴とする請求項２６または２７に記載のバージョン管理方法。

【請求項２９】　前記バージョン管理ステップは、サポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルと比較して判定することを特徴とする請求項２８に記載のバージョン管理方法。

【請求項３０】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションが使用する全関数についてサポートが可能か否かを判定するバージョン管理方法であって、

前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と

、前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップ

を含むことを特徴とするバージョン管理方法。

【請求項３１】　画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションが使用する全関数についてサポートが可能か否かを判定するバージョン管理方法であって、

前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップ

を含むことを特徴とするバージョン管理方法。

【請求項３２】　前記バージョン管理ステップは、サポート可能な範囲のバージョン差か否かを前記アプリケーションが使用する全ての関数番号とバージョン番号とに基づいて判定することを特徴とする請求項３０または３１に記載のバージョン管理方法。

【請求項３３】　前記バージョン管理ステップは、前記アプリケーションが使用する全ての関数番号とバージョン番号を、サポート可能な範囲の関数番号とバージョン番号が登録されたバージョン管理テーブルと比較して判定することを特徴とする請求項３２に記載のバージョン管理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明は、コピー、プリンタ、スキャナおよびファクシミリなどの画像形成処理にかかるユーザサービスを提供する画像形成装置に搭載されるアプリケーションとの間でバージョン差が生じた場合に、バージョン不整合による不具合発生を防止することができる画像形成装置およびラッピング処理方法並びにバージョン管理方法に関するものである。

【０００２】

【従来の技術】

近年では、プリンタ、コピー、ファクシミリ、スキャナなどの各装置の機能を１つの筐体内に収納した画像形成装置（以下、「複合機」という。）が一般的に知られている。この複合機は、１つの筐体内に表示部、印刷部および撮像部などを設けるとともに、プリンタ、コピーおよびファクシミリ装置にそれぞれ対応する３種類のソフトウェアを設け、これらのソフトウェアを切り替えることによって、当該装置をプリンタ、コピー、スキャナまたはファクシミリ装置として動作させるものである。

【０００３】

このような従来の複合機では、プリンタ、コピー、ファクシミリ、スキャナなどの各機能単位でアプリケーションプログラムが起動され、ハードウェア資源にアクセスする機能を持ち合わせている。その際、アプリケーションプログラムとオペレーティングシステム（ＯＳ）のバージョンが同じことが前提となるが、例えば、ＯＳをバージョンアップしてアプリケーションとの間でバージョン差が生じた場合、今まで使えていた機能が使えなくなったり、アプリケーションそのものが起動しなくなったりすることがある。

【０００４】

このため、従来の複合機では、ＯＳなどをバージョンアップすると、それに伴ってアプリケーションの方もコンパイルし直すことで、常に対応したバージョン関係にあることが要請されている。

【０００５】

【発明が解決しようとする課題】

ところで、このような従来の複合機では、プリンタ、コピー、スキャナおよびファクシミリ装置に対応するソフトウェアがそれぞれ別個に設けられているため、各ソフトウェアの開発に多大の時間を要する。このため、出願人は、表示部、印刷部および撮像部などの画像形成処理で使用されるハードウェア資源を有し、プリンタ、コピーまたはファクシミリなどの各ユーザサービスにそれぞれ固有の処理を行うアプリケーションを複数搭載し、これらのアプリケーションとハードウェア資源との間に介在して、ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とするハードウェア資源の管理、実行制御並びに画像形成処理を行う各種コントロールサービスからなるプラットホームを備えた画像形成装置（複合機）を発明した。

【０００６】

このような新規な複合機では、アプリケーションとコントロールサービスとを別個に設けているため、複合機の出荷後にユーザもしくは第三者であるサードベンダが新規なアプリケーションを開発して複合機に搭載することが可能となり、これによって多種多様な機能を提供することが可能となる。

【０００７】

このような新規な複合機では、アプリケーションの少なくとも２つが共通的に必要とするサービスを提供するコントロールサービスをアプリケーションと別個に設けた構成となっているため、新規アプリケーションを開発する場合には、各種コントロールサービスとのプロセス間通信を実現する処理をソースコードで記述する必要がある。しかしながら、新規アプリケーションを開発する場合、各コントロールサービスが提供する関数やメッセージなどを正確に把握した上で、予め規定された手順で記述しなければならず、各コントロールサービスやアプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）をデバッグや機能追加によってバージョンアップが繰り返されると、ベンダーはどのバージョンに合わせてアプリケーション開発すればよいかが非常に分かり難くなり、アプリケーションの開発自体を阻害するおそれがある。このことは、固定的な複数の機能を寄せ集めた従来の複合機では問題にならなかった新規な課題である。

【０００８】

また、各コントロールサービスとアプリケーションのインタフェース（ＡＰＩ）のすべてを、新規アプリケーションを開発するサードベンダなどの第三者に開示することは、プログラムの秘匿性の面から言って好ましくない。特に、複合機のシステムに大きく影響を与えるような箇所については、第三者に対して隠蔽することで直接コントロールサービスに対してアクセスされるのを回避することが可能となり、複合機のセキュリティ面および障害発生を未然に防止する面からも好ましい。

【０００９】

この発明は上記に鑑みてなされたもので、多数のプロセスが実行される画像形成装置上で動作するアプリケーションとコントロールサービスおよびインタフェースとの間でバージョン管理を行い、バージョン差が生じた場合でもバージョン差を吸収して整合性をとるようにすると共に、アプリケーションとコントロールサービス間の複雑なインタフェースを隠蔽して重要なインタフェースの秘匿性を確保すると共に、特定のインタフェースを開示することによって新規アプリケーションの開発効率を向上させることができる画像形成装置およびラッピング処理方法並びにバージョン管理方法を得ることを目的とする。

【００１０】

【課題を解決するための手段】

上記目的を達成するため、請求項１にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記コントロールサービスのいずれかのコンポーネントがバージョンアップしたときに、バージョン差の生じたアプリケーショ

ンからの関数呼び出しに対して整合性をとるラッピング処理手段を備えたことを特徴とする。

【００１１】

この請求項１にかかる発明によれば、コントロールサービスのいずれかのコンポーネントがバージョンアップしたとき、ラッピング処理手段によってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるようにしたため、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【００１２】

また、請求項２にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記アプリケーションプログラムインタフェースがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理手段を備えたことを特徴とする。

【００１３】

この請求項２にかかる発明によれば、アプリケーションプログラムインタフェースがバージョンアップしたとき、ラッピング処理手段によってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるようにしたため、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【００１４】

また、請求項３にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケ

ーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記コントロールサービスから前記アプリケーションに対するメッセージを取捨選択するラッピング処理手段を備えたことを特徴とする。

【００１５】

この請求項３にかかる発明によれば、ラッピング処理手段によってコントロールサービスからアプリケーションに対するメッセージを取捨選択するため、サードベンダなどの第三者に対してはコントロールサービスとの間のインタフェースを隠蔽し、重要なインタフェースの秘匿性を確保して、第三者からコントロールサービスに直接アクセスされるのを回避することにより、画像形成装置におけるセキュリティ面が確保され、障害発生を未然に防止することができる。

【００１６】

また、請求項４にかかる発明は、請求項１または２に記載の画像形成装置において、前記ラッピング処理手段は、関数または関数の引数が追加されてバージョンアップしたときに、追加された関数または関数の引数に対応する機能を無視して整合性をとることを特徴とする。

【００１７】

この請求項４にかかる発明によれば、関数または関数の引数が追加されてバージョンアップした場合、ラッピング処理手段はアプリケーションに対して提供する関数または関数の引数を変更しなければ、追加された機能は使えないが既存の機能はそのまま使えるため、追加機能を無視すればアプリケーションを再コンパイルすることなくバージョン差を吸収することができる。

【００１８】

また、請求項５にかかる発明は、請求項１または２に記載の画像形成装置において、前記ラッピング処理手段は、関数または関数の引数が分割されてバージョ

ンアップしたときに、分割された関数または関数の引数に応じてダミーの関数または関数の引数を補填して整合性をとることを特徴とする。

【００１９】

この請求項５にかかる発明によれば、関数または関数の引数が分割されてバージョンアップした場合、対応する関数や関数の引数が増えるため、そのままでは関数呼び出しを行うことができず、ラッピング処理手段によって増加した関数や関数の引数に対応するダミーの関数または関数の引数を補填することによって整合性をとることができ、バージョン差を吸収することができる。

【００２０】

また、請求項６にかかる発明は、請求項３に記載の画像形成装置において、前記ラッピング処理手段は、前記アプリケーションに対して通知しない特定のメッセージを予め設定可能にしたことを特徴とする。

【００２１】

この請求項６にかかる発明によれば、特定のメッセージをアプリケーションに対して通知しないように、ラッピング処理手段に対して予め設定することが可能なため、重要なインタフェースの秘匿性を確保することができる。

【００２２】

また、請求項７にかかる発明は、請求項１～６のいずれか一つに記載の画像形成装置において、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスをさらに備え、前記ラッピング処理手段は、前記仮想アプリケーションサービスに含まれていることを特徴とする。

【００２３】

この請求項７にかかる発明によれば、ラッピング処理手段は仮想アプリケーションサービスに含まれており、この仮想アプリケーションサービスは、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作するため、バージョン差が生じた場合にバージョン差を吸収して整合性をとることができ、また、コン

トロールサービスからアプリケーションに対するメッセージを選択することによって、インタフェースを隠蔽して重要なインタフェースの秘匿性を確保しつつ、新規アプリケーションのプログラム開発を効率的に行うことができる。

【００２４】

また、請求項８にかかる発明は、請求項７に記載の画像形成装置において、前記ラッピング処理手段は、前記仮想アプリケーションサービスのプロセス内部でスレッドとして生成されることを特徴とする。

【００２５】

この請求項８にかかる発明によれば、ラッピング処理手段は、仮想アプリケーションサービスのプロセス内部にスレッドとして生成され、実行されるため、仮想アプリケーションサービスのプロセス内部で軽快に動作させることができると共に、プロセス内部に複数個のスレッドを生成した場合であっても、それらを並行処理することができる。

【００２６】

また、請求項９にかかる発明は、請求項１～８のいずれか一つに記載の画像形成装置において、前記アプリケーションの実行時に、共通の関数を利用して他のプログラムと結合させて実行形式のプログラムを作成する動的リンク手段をさらに備えていることを特徴とする。

【００２７】

この請求項９にかかる発明によれば、アプリケーションの実行時に、共通の関数を利用して他のプログラムと結合させることにより実行形式のプログラムを作成する動的リンク手段を備えているため、アプリケーションとコントロールサービスあるいはインタフェース間のバージョン差の吸収力が増大し、整合性が一層とれるようになると共に、インタフェースを隠蔽して重要なインタフェースの秘匿性をより強固に確保することができる。

【００２８】

また、請求項１０にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間

に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記アプリケーションと前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段を備えたことを特徴とする。

【００２９】

この請求項１０にかかる発明によれば、アプリケーションとコントロールサービスのいずれかのコンポーネントとの間でバージョン差を生じた場合に、そのバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたため、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行前にバージョンの適合性の有無が判断できることから、障害の発生を未然に防止することができる。

【００３０】

また、請求項１１にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記アプリケーションと前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段を備えたことを特徴とする。

【００３１】

この請求項１１にかかる発明によれば、アプリケーションとアプリケーションプログラムインタフェースとの間でバージョン差が生じた場合に、そのバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたため、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行前にバージョンの適合性の有無が判断できることから、障害の発生を未然に防止することができる。

【００３２】

また、請求項１２にかかる発明は、請求項１０または１１に記載の画像形成装置において、前記バージョン管理手段は、サポート可能な範囲のバージョンか否かを前記アプリケーションのバージョン番号に基づいて判定することを特徴とする。

【００３３】

この請求項１２にかかる発明によれば、バージョン管理手段がサポート可能な範囲のバージョンか否かをアプリケーションのバージョン番号に基づいて判定するようにしたため、容易かつ正確に判定することができる。

【００３４】

また、請求項１３にかかる発明は、請求項１２に記載の画像形成装置において、前記バージョン管理手段は、サポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルをさらに備え、前記アプリケーションのバージョン番号を前記バージョン管理テーブルと比較して判定することを特徴とする。

【００３５】

この請求項１３にかかる発明によれば、バージョン管理手段がサポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルを備えていて、アプリケーションのバージョン番号をバージョン管理テーブルと比較して判定するようにしたため、サポート可能なアプリケーションか否かを容易かつ正確に判定することができる。

【００３６】

また、請求項１４にかかる発明は、画像形成処理で使用されるハードウェア資

源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段を備えたことを特徴とする。

【００３７】

この請求項１４にかかる発明によれば、バージョン管理手段によりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、コントロールサービスとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたため、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理手段によるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【００３８】

また、請求項１５にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を

受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置であって、前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理手段を備えたことを特徴とする。

【００３９】

この請求項１５にかかる発明によれば、バージョン管理手段によりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、アプリケーションプログラムインタフェースとのバージョンとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたため、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理手段によるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【００４０】

また、請求項１６にかかる発明は、請求項１４または１５に記載の画像形成装置において、前記バージョン管理手段は、サポート可能な範囲のバージョンか否かを前記アプリケーションが使用する全ての関数番号とバージョン番号とに基づいて判定することを特徴とする。

【００４１】

この請求項１６にかかる発明によれば、バージョン管理手段がサポート可能な範囲のバージョンか否かを判定するにあたって、アプリケーションが使用する全ての関数の番号とバージョン番号に基づいて行われるため、単一な番号比較だけでできることから管理が容易となり、バージョンチェック処理の負荷が小さくなるという利点がある。

【００４２】

また、請求項１７にかかる発明は、請求項１６に記載の画像形成装置において

、前記バージョン管理手段は、サポート可能な範囲の全ての関数番号とバージョン番号とが登録されたバージョン管理テーブルをさらに備え、前記アプリケーションが使用する全ての関数番号とバージョン番号を前記バージョン管理テーブルと比較して判定することを特徴とする。

【００４３】

この請求項１７にかかる発明によれば、バージョン管理手段は、バージョン管理テーブルに登録されているサポート可能な範囲の全ての関数番号とバージョン番号と、アプリケーションが使用する全ての関数番号とバージョン番号とを比較して判定するようにしたため、バージョン差の生じたアプリケーションがサポート可能か否かを容易かつ正確に判定することができる。

【００４４】

また、請求項１８にかかる発明は、請求項１０～１７のいずれか一つに記載の画像形成装置において、前記コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつ前記アプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスをさらに備え、前記バージョン管理手段は、前記仮想アプリケーションサービスに含まれていることを特徴とする。

【００４５】

この請求項１８にかかる発明によれば、バージョン管理手段は、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスに含まれているため、バージョン差が生じた場合にバージョン差を吸収して整合性をとることができ、また、コントロールサービスからアプリケーションに対するメッセージを選択することによって、インタフェースを隠蔽して重要なインタフェースの秘匿性を確保しつつ、新規アプリケーションのプログラム開発を効率的に行うことができる。

【００４６】

また、請求項１９にかかる発明は、請求項１８に記載の画像形成装置において、前記バージョン管理手段は、前記仮想アプリケーションサービスのプロセス内

部でスレッドとして生成されることを特徴とする。

【００４７】

この請求項１９にかかる発明によれば、バージョン管理手段は、仮想アプリケーションサービスのプロセス内部にスレッドとして生成され、実行されるため、仮想アプリケーションサービスのプロセス内部で軽快に動作させることができると共に、プロセス内部に複数個のスレッドを生成した場合であっても、それらを並行処理することができる。

【００４８】

また、請求項２０にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとのバージョン差を吸収するラッピング処理方法であって、前記コントロールサービスのいずれかのコンポーネントがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理ステップを含むことを特徴とする。

【００４９】

この請求項２０にかかる発明によれば、コントロールサービスのいずれかのコンポーネントがバージョンアップしたとき、ラッピング処理ステップによってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるため、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【００５０】

また、請求項２１にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリ

ケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとのバージョン差を吸収するラッピング処理方法であって、前記アプリケーションプログラムインタフェースがバージョンアップしたときに、バージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるラッピング処理ステップを含むことを特徴とする。

【００５１】

この請求項２１にかかる発明によれば、アプリケーションプログラムインタフェースがバージョンアップしたとき、ラッピング処理ステップによってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるようにしたため、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【００５２】

また、請求項２２にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数によって前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとのバージョン差を吸収するラッピング処理方法であって、前記コントロールサービスから前記アプリケーションに対するメッセージを取捨選択するラッピング処理ステップを含むことを特徴とする。

【００５３】

この請求項２２にかかる発明によれば、ラッピング処理ステップによってコントロールサービスからアプリケーションに対するメッセージを取捨選択するため、サードベンダなどの第三者に対してコントロールサービスとの間のインタフェースを隠蔽することで、重要なインタフェースの秘匿性を確保し、第三者からコントロールサービスに直接アクセスされるのを回避して、画像形成装置におけるセキュリティ面を確保することにより、障害の発生が未然に防止できる。

【００５４】

また、請求項２３にかかる発明は、請求項２０または２１に記載のラッピング処理方法において、前記ラッピング処理ステップは、関数または関数の引数が追加されてバージョンアップしたときに、追加された関数または関数の引数に対応する機能を無視して整合性をとることを特徴とする。

【００５５】

この請求項２３にかかる発明によれば、ラッピング処理ステップにおいて、関数または関数の引数が追加されてバージョンアップした場合、追加された関数または関数の引数に対応する機能を無視して整合性をとるようにしたため、アプリケーションからの関数呼び出しに使われる関数または関数の引数をそのまま使うことができ、アプリケーションを再コンパイルすることなくバージョン差を吸収することができる。

【００５６】

また、請求項２４にかかる発明は、請求項２０または２１に記載のラッピング処理方法において、前記ラッピング処理ステップは、関数または関数の引数が分割されてバージョンアップしたときに、分割された関数または関数の引数に応じてダミーの関数または関数の引数を補填して整合性をとることを特徴とする。

【００５７】

この請求項２４にかかる発明によれば、ラッピング処理ステップにおいて、関数または関数の引数が分割されてバージョンアップした場合、分割された関数または関数の引数に応じてダミーの関数または関数の引数を補填して整合性をとるようにしたため、関数の対応関係を維持することが可能となり、アプリケーションを再コンパイルすることなくバージョン差を吸収することができる。

【００５８】

また、請求項２５にかかる発明は、請求項２２に記載のラッピング処理方法において、前記ラッピング処理ステップは、前記アプリケーションに対して通知しない特定のメッセージを予め設定可能にしたことを特徴とする。

【００５９】

この請求項２５にかかる発明によれば、ラッピング処理ステップにおいて、アプリケーションに対して通知しない特定のメッセージが予め設定可能なため、サードベンダなどに公開するインタフェースの基準を任意に決定することができ、重要なインタフェースの秘匿性を確保することができる。

【００６０】

また、請求項２６にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとの間のバージョンを管理するバージョン管理方法であって、前記アプリケーションと前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップを含むことを特徴とする。

【００６１】

この請求項２６にかかる発明によれば、アプリケーションとコントロールサービスのいずれかのコンポーネントとの間でバージョン差が生じた場合に、バージョン管理ステップによりバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたため、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行

前にバージョンの適合性の有無が判断でき、障害の発生を未然に防止することができる。

【００６２】

また、請求項２７にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションとの間のバージョンを管理するバージョン管理方法であって、前記アプリケーションと前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップを含むことを特徴とする。

【００６３】

この請求項２７にかかる発明によれば、アプリケーションとアプリケーションプログラムインタフェースとの間でバージョン差が生じた場合に、バージョン管理ステップによりバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたため、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行前にバージョンの適合性の有無が判断でき、障害の発生を未然に防止することができる。

【００６４】

また、請求項２８にかかる発明は、請求項２６または２７に記載のバージョン管理方法において、前記バージョン管理ステップは、サポート可能な範囲のバージョン差か否かを前記アプリケーションのバージョン番号に基づいて判定することを特徴とする。

【００６５】

この請求項２８にかかる発明によれば、バージョン管理ステップにおいて、サポート可能な範囲のバージョンか否かをアプリケーションのバージョン番号に基づいて判定するようにしたため、容易かつ正確に判定することができる。

【００６６】

また、請求項２９にかかる発明は、請求項２８に記載のバージョン管理方法において、前記バージョン管理ステップは、サポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルと比較して判定することを特徴とする。

【００６７】

この請求項２９にかかる発明によれば、バージョン管理ステップにおいて、サポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルを用い、登録されるアプリケーションのバージョン番号をバージョン管理テーブルと比較して判定するようにしたため、サポート可能なアプリケーションか否かを容易かつ正確に判定することができる。

【００６８】

また、請求項３０にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションが使用する全関数についてサポートが可能か否かを判定するバージョン管理方法であって、前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記コントロールサービスのいずれかのコンポーネントとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップを含むことを特徴とする。

【００６９】

この請求項３０にかかる発明によれば、バージョン管理ステップによりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、コントロールサービスとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたため、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理ステップによるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【００７０】

また、請求項３１にかかる発明は、画像形成処理で使用されるハードウェア資源と、画像形成処理にかかるユーザサービスにそれぞれ固有の処理を行うアプリケーションのプロセスと、前記アプリケーションと前記ハードウェア資源との間に介在し、前記ユーザサービスを提供する際に、アプリケーションの少なくとも２つが共通的に必要とする前記ハードウェア資源の獲得要求、管理、実行制御並びに画像形成処理を行うコントロールサービスのプロセスと、前記コントロールサービスが予め定義された関数により前記アプリケーションからの処理要求を受信可能とするアプリケーションプログラムインタフェースとを備えた画像形成装置上で、前記アプリケーションが使用する全関数についてサポートが可能か否かを判定するバージョン管理方法であって、前記アプリケーションが使用する関数を全種類抽出し、その抽出した各関数と、前記アプリケーションプログラムインタフェースとの間にバージョン差が生じたときに、そのバージョン差がサポート可能な範囲か否かを判定して、その判定結果を前記アプリケーションに通知するバージョン管理ステップを含むことを特徴とする。

【００７１】

この請求項３１にかかる発明によれば、バージョン管理ステップによりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、アプリケーションプログラムインタフェースとのバージョンとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果を

アプリケーションに通知するようにしたため、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理ステップによるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【００７２】

また、請求項３２にかかる発明は、請求項３０または３１に記載のバージョン管理方法において、前記バージョン管理ステップは、サポート可能な範囲のバージョン差か否かを前記アプリケーションが使用する全ての関数番号とバージョン番号とに基づいて判定することを特徴とする。

【００７３】

この請求項３２にかかる発明によれば、バージョン管理ステップによりサポート可能な範囲のバージョンか否かを判定するにあたって、アプリケーションが使用する全ての関数の番号とバージョン番号に基づいて行われるため、単一な番号比較だけでできることから管理が容易となり、バージョンチェック処理の負荷が小さくなるという利点がある。

【００７４】

また、請求項３３にかかる発明は、請求項３２に記載のバージョン管理方法において、前記バージョン管理ステップは、前記アプリケーションが使用する全ての関数番号とバージョン番号を、サポート可能な範囲の関数番号とバージョン番号が登録されたバージョン管理テーブルと比較して判定することを特徴とする。

【００７５】

この請求項３３にかかる発明によれば、バージョン管理ステップでは、バージョン管理テーブルに登録されているサポート可能な範囲の全ての関数番号とバージョン番号と、アプリケーションが使用する全ての関数番号とバージョン番号とを比較して判定するようにしたため、バージョン差の生じたアプリケーションがサポート可能か否かを容易かつ正確に判定することができる。

【００７６】

【発明の実施の形態】

以下に添付図面を参照して、この発明にかかる画像形成装置およびラッピング処理方法並びにバージョン管理方法の好適な実施の形態を詳細に説明する。

【００７７】

（実施の形態１）

図１は、この発明の実施の形態１である画像形成装置（以下、「複合機」という）の構成を示すブロック図である。図１に示すように、複合機１００は、白黒レーザプリンタ（B&W LP）１０１と、カラーレーザプリンタ（Color LP）１０２と、スキャナ、ファクシミリ、ハードディスク、メモリ、ネットワークインタフェースなどのハードウェアリソース１０３を有するとともに、プラットホーム１２０とアプリケーション１３０と仮想アプリケーションサービス（ＶＡＳ：Ｖｉｒｔｕａｌ　Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｉｃｅ）１４０から構成されるソフトウェア群１１０とを備えている。

【００７８】

仮想アプリケーションサービス（ＶＡＳ）１４０は、アプリケーション１３０とプラットホーム１２０の間に配置される。ＶＡＳ１４０は、アプリケーション１３０の各アプリが初めて登録されるときに、登録処理が行われ、アプリから見るとプラットホーム１２０のサービス層として認識され、サービス層から見るとアプリとして認識されるように登録される。このＶＡＳ１４０の第１の基本機能としては、プラットホーム１２０の各種コントロールサービスあるいはそのアプリケーションプログラムインタフェース（ＡＰＩ）が仕様変更などでバージョンアップされた場合に、バージョン差の生じたアプリケーション１３０を再コンパイルすることなくそのまま関数呼び出しが行えるよう、バージョン差を吸収して整合性を保つと共に、コントロールサービスからのメッセージを取捨選択することによってプラットホーム１２０を意図的に隠蔽することができるラッピング機能を備えている。

【００７９】

また、このＶＡＳ１４０の第２の基本機能としては、各アプリが登録される際に、各アプリとコントロールサービスおよびＡＰＩのそれぞれのバージョンを検出し、その検出されたバージョン差がＶＡＳ１４０によってサポート可能な範囲

か否かを判定して、その判定結果をアプリに通知することにより、アプリ動作前にバージョンが不整合かどうかが判明するバージョン管理機能を備えている。

【００８０】

プラットホーム１２０は、アプリケーションからの処理要求を解釈してハードウェア資源の獲得要求を発生させるコントロールサービスと、一または複数のハードウェア資源の管理を行い、コントロールサービスからの獲得要求を調停するシステムリソースマネージャ（ＳＲＭ）１２３と、汎用ＯＳ１２１とを有する。

【００８１】

コントロールサービスは、複数のサービスモジュールから形成され、ＳＣＳ（システムコントロールサービス）１２２と、ＥＣＳ（エンジンコントロールサービス）１２４と、ＭＣＳ（メモリコントロールサービス）１２５と、ＯＣＳ（オペレーションパネルコントロールサービス）１２６と、ＦＣＳ（ファックスコントロールサービス）１２７と、ＮＣＳ（ネットワークコントロールサービス）１２８とから構成される。なお、このプラットホーム１２０は、予め定義された関数により前記アプリケーション１３０から処理要求を受信可能とするアプリケーションプログラムインタフェース（ＡＰＩ）を有している。

【００８２】

汎用ＯＳ１２１は、ＵＮＩＸ（登録商標）などの汎用オペレーティングシステムであり、プラットホーム１２０並びにアプリケーション１３０の各ソフトウェアをそれぞれプロセスとして並列実行する。

【００８３】

ＳＲＭ１２３のプロセスは、ＳＣＳ１２２とともにシステムの制御およびリソースの管理を行うものである。ＳＲＭ１２３のプロセスは、スキャナ部やプリンタ部などのエンジン、メモリ、ＨＤＤファイル、ホストＩ／Ｏ（セントロＩ／Ｆ、ネットワークＩ／Ｆ、ＩＥＥＥ１３９４　Ｉ／Ｆ、ＲＳ２３２Ｃ　Ｉ／Ｆなど）のハードウェア資源を利用する上位層からの要求にしたがって調停を行い、実行制御する。

【００８４】

具体的には、このＳＲＭ１２３は、要求されたハードウェア資源が利用可能で

あるか（他の要求により利用されていないかどうか）を判断し、利用可能であれば要求されたハードウェア資源が利用可能である旨を上位層に伝える。また、ＳＲＭ１２３は、上位層からの要求に対してハードウェア資源の利用スケジューリングを行い、要求内容（例えば、プリンタエンジンにより紙搬送と作像動作、メモリ確保、ファイル生成など）を直接実施している。

【００８５】

ＳＣＳ１２２のプロセスは、アプリ管理、操作部制御、システム画面表示、ＬＥＤ表示、リソース管理、割り込みアプリ制御などを行う。

【００８６】

ＥＣＳ１２４のプロセスは、白黒レーザプリンタ（B&W LP）１０１、カラーレーザプリンタ（Color LP）１０２、スキャナ、ファクシミリなどからなるハードウェアリソース１０３のエンジンの制御を行う。

【００８７】

ＭＣＳ１２５のプロセスは、画像メモリの取得および解放、ハードディスク装置（ＨＤＤ）の利用、画像データの圧縮および伸張などを行う。

【００８８】

ＦＣＳ１２７のプロセスは、システムコントローラの各アプリ層からＰＳＴＮ／ＩＳＤＮ網を利用したファクシミリ送受信、ＢＫＭ（バックアップＳＲＡＭ）で管理されている各種ファクシミリデータの登録／引用、ファクシミリ読みとり、ファクシミリ受信印刷、融合送受信を行うためのＡＰＩを提供する。

【００８９】

ＮＣＳ１２８のプロセスは、ネットワークＩ／Ｏを必要とするアプリケーションに対して共通に利用できるサービスを提供するためのプロセスであり、ネットワーク側から各プロトコルによって受信したデータを各アプリケーションに振り分けたり、アプリケーションからデータをネットワーク側に送信する際の仲介を行う。具体的には、ftpd、httpd、lpd、snmpd、telnetd、smtpdなどのサーバデーモンや、同プロトコルのクライアント機能などを有している。

【００９０】

ＯＣＳ１２６のプロセスは、オペレータ（ユーザ）と本体制御間の情報伝達手

段となるオペレーションパネル（操作パネル）の制御を行う。ＯＣＳ１２６は、オペレーションパネルからキー押下をキーイベントとして取得し、取得したキーに対応したキーイベント関数をＳＣＳ１２２に送信するＯＣＳプロセスの部分と、アプリケーション１３０またはコントロールサービスからの要求によりオペレーションパネルに各種画面を描画出力する描画関数やその他オペレーションパネルに対する制御を行う関数などが予め登録されたＯＣＳライブラリの部分とから構成される。このＯＣＳライブラリは、アプリケーション１３０およびコントロールサービスの各モジュールにリンクされて実装されている。なお、ＯＣＳ１２６のすべてをプロセスとして動作させるように構成しても良く、あるいはＯＣＳ１２６のすべてをＯＣＳライブラリとして構成しても良い。

【００９１】

アプリケーション１３０は、ページ記述言語（ＰＤＬ）、ＰＣＬおよびポストスクリプト（ＰＳ）を有するプリンタ用のアプリケーションであるプリンタアプリ１１１と、コピー用アプリケーションであるコピーアプリ１１２と、ファクシミリ用アプリケーションであるファックスアプリ１１３と、スキャナ用アプリケーションであるスキャナアプリ１１４と、ネットワークファイル用アプリケーションであるネットファイルアプリ１１５と、工程検査用アプリケーションである工程検査アプリ１１６とを有している。これらの各アプリは、その起動時にＶＡＳ１４０に対して自プロセスのプロセスＩＤとともにアプリ登録要求メッセージを送信し、アプリ登録要求メッセージを受信したＶＡＳ１４０によって、起動したアプリに対する登録が行われるようになっている。

【００９２】

アプリケーション１３０の各プロセス、コントロールサービスの各プロセスは、関数呼び出しとその戻り値送信およびメッセージの送受信によってプロセス間通信を行いながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを実現している。

【００９３】

このように、実施の形態１にかかる複合機１００には、複数のアプリケーション１３０および複数のコントロールサービスが存在し、いずれもプロセスとして

動作している。そして、これらの各プロセス内部には、一または複数のスレッドが生成されて、スレッド単位の並列実行が行われる。そして、コントロールサービスがアプリケーション１３０に対し共通サービスを提供しており、このため、これらの多数のプロセスが並列動作、およびスレッドの並列動作を行って互いにプロセス間通信を行って協調動作をしながら、コピー、プリンタ、スキャナ、ファクシミリなどの画像形成処理にかかるユーザサービスを提供するようになっている。また、複合機１００には、サードベンダなどの第三者がコントロールサービス層の上のアプリケーション層に新規アプリ１１７、１１８を開発して搭載することが可能となっている。図１では、この新規アプリ１１７、１１８を搭載した例を示している。

【００９４】

なお、実施の形態１にかかる複合機１００では、複数のアプリケーション１３０のプロセスと複数のコントロールサービスのプロセスとが動作しているが、アプリケーション１３０とコントロールサービスのプロセスをそれぞれ単一の構成とすることも可能である。また、各アプリケーション１３０は、アプリケーションごとに追加または削除することができる。さらに、実施の形態１にかかる複合機１００では、ＶＡＳ１４０に加えて動的リンク手段としてのダイナミックリンクライブラリ（ＤＬＬ）を採用することが可能である。

【００９５】

図２は、実施の形態１にかかる複合機１００のＶＡＳ１４０の構成と、ＶＡＳ１４０と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図２では、アプリケーション１３０の例として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８を示しているが、他のアプリでも同様の構成である。

【００９６】

仮想アプリケーションサービス（ＶＡＳ）１４０のプロセスには、ディスパッチャ１４４と、制御スレッド１４３と、ラッピング処理スレッド１４１と、バージョン管理スレッド１４２とが動作している。

【００９７】

ディスパッチャ１４４は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じて制御スレッド１４３、ラッピング処理スレッド１４１、バージョン管理スレッド１４２に処理要求を行うものである。実施の形態１の複合機１００では、ディスパッチャ１４４は、各アプリが起動したときにアプリからのアプリ登録要求メッセージを受信すると、そのアプリ登録要求メッセージを制御スレッド１４３に送信するようになっている。

【００９８】

制御スレッド１４３は、ディスパッチャ１４４からアプリ登録要求メッセージを受信してアプリ登録処理を行う。ここで、アプリ登録処理とは、ＲＡＭ２１０にアプリ登録テーブル（図示せず）を生成し、アプリ登録要求メッセージを送信したアプリの識別情報であるアプリＩＤをアプリ登録テーブルに記録する処理をいう。

【００９９】

また、制御スレッド１４３は、ＨＤ２００に格納されたラッピング処理情報ファイル２０１を参照して、アプリ登録要求を行ったアプリについて、ラッピング処理情報が記録されているか否かをチェックする。

【０１００】

そして、ラッピング処理手段としてのラッピング処理スレッド１４１は、コントロールサービスあるいはその上のＡＰＩの関数または関数の引数が追加されてバージョンアップした場合、制御スレッド１４３からの処理要求を受け、アプリからの関数呼び出しに対して対応する関数内で実行されるため、追加された機能を無視する（使えないとする）ならばバージョンの不整合は起こらない。これは、ＶＡＳ１４０上に設けられた不図示のインタフェース（ＶＡＳ－ＡＰＩ）は、コントロールサービスあるいはその上のＡＰＩをバージョンアップしたとしてもアプリ間での整合性が保持されるため、アプリを再コンパイルする必要はなくなる。

【０１０１】

また、コントロールサービスあるいはその上のＡＰＩの関数または関数の引数

が分割されてバージョンアップした場合は、ラッピング処理情報ファイル２０１を参照して対応するダミーの関数または関数の引数を補填することにより、アプリからの関数呼び出しに対して対応関数を起動させることが可能となり、ＶＡＳ１４０がバージョン差を吸収することから、アプリを再コンパイルする必要がなくなる。

【０１０２】

さらに、ラッピング処理スレッド１４１は、コントロールサービス層１５０からアプリに対して行われるメッセージ通知を選択的に行えるよう、予めラッピング処理情報ファイル２０１に設定することが可能である。特に、複合機１００のシステムに大きく影響を与えるインタフェースについては、サードベンダなどの第三者に対し隠蔽しておき、直接コントロールサービスに対してアクセスすることを回避することが、複合機のセキュリティ面および障害発生の未然防止の面から好ましい。

【０１０３】

また、バージョン管理手段としてのバージョン管理スレッド１４２は、制御スレッド１４３からの処理要求を受け、登録されたアプリのバージョンとコントロールサービスのいずれかのコンポーネントおよびＡＰＩとのバージョンの違いを検出すると、サポート可能な範囲のバージョン番号などが登録されているＲＡＭ２１０のバージョン管理テーブル２１１と比較することにより、発生したバージョン差がＶＡＳ１４０によってサポート可能な範囲か否かを判定することができる。かかるサポート可能な範囲のバージョン番号などが登録されたバージョン管理情報は、アプリごとのレコードとして記録される。

【０１０４】

図３は、ＨＤ２００に格納されるラッピング処理情報ファイル２０１の内容例を示す説明図である。図３に示すように、ラッピング処理情報ファイル２０１は、コントロールサービス層１５０からアプリに対するメッセージ（イベント）の非通知設定を行うための情報ファイルであり、複合機１００のシステムに大きく影響を与えるインタフェースについては第三者に対して隠蔽するよう、メッセージごとに任意の非通知設定を行うことができる。図３のイベントＡとＣは、非通

知設定がなされているのでアプリには通知されず、また、イベントＢのように非通知設定がなされていないイベントについては、通常通り通知が行われる。

【０１０５】

図４は、ＲＡＭ２１０に格納される一括管理用のバージョン管理テーブル２１１の内容例を示す説明図である。図４に示すように、バージョン管理テーブル２１１には、ＶＡＳ１４０でサポート範囲内とする各アプリ（アプリＩＤ）のバージョン番号が格納されていて、アプリ登録の際に通知されるアプリのバージョン番号をこのバージョン管理テーブル２１１に格納されるバージョン番号と比較することにより、サポート範囲内か否かを容易に判定することができる。ここでいうサポート範囲とは、ＶＡＳ１４０のリリース時点より前のバージョンのことであり、ＶＡＳ１４０によって吸収可能な一定のバージョン範囲のことをいう。

【０１０６】

図５は、ＲＡＭ２１０に格納される関数単位管理用のバージョン管理テーブル２１１の内容例を示す説明図である。図５に示すように、このバージョン管理テーブル２１１には、関数番号とバージョン番号とそのサポート範囲とが格納されていて、アプリがＶＡＳ１４０に対して使用する全種類の関数を抽出し、各関数をバージョン管理テーブル２１１と比較することにより、関数単位でサポートが可能か否かを判定することができる。

【０１０７】

次に、このように構成された複合機１００のＶＡＳ１４０によるラッピング処理とバージョン管理について説明する。図６は、ＶＡＳ１４０の制御スレッド１４３によりバージョン差を吸収するためのラッピング処理を行う手順を示すフローチャートである。

【０１０８】

図６において、アプリとコントロールサービスおよびそのＡＰＩのバージョンが一致している間は、障害は発生しないが、コントロールサービスあるいはそのＡＰＩがバージョンアップされてアプリとの間にバージョン差が生じた場合、バージョン不整合が問題となる。ここで、アプリのバージョンとは、アプリのコンパイル時に使用したＶＡＳヘッダファイルに定義されたバージョンのことをいう

。そこで、ステップＳ６０１において、コントロールサービスおよびそのＡＰＩがバージョンアップされた場合、そのバージョンアップの態様を判別する（ステップＳ６０２）。例えば、関数または関数の引数の追加、削除、変更、分割、統合などによるバージョンアップが考えられるが、関数または関数の引数が削除、変更、統合された場合は、対応する関数が存在しなくなるため、アプリからの関数呼び出しがあっても整合性をとることができない。このため、アプリに対してＮＧであることを通知し（ステップＳ６０６）、終了処理する（ステップＳ６０７）。

【０１０９】

一方、ステップＳ６０２において、関数または関数の引数の追加あるいは分割によるバージョンアップの場合、さらに追加か分割かを判別して（ステップＳ６０３）、追加の場合は既存の関数自体は変化しないことから、追加された機能は使えないが、関数呼び出しによるバージョンの整合性を保つことができるので（ステップＳ６０８）、そのまま通常に処理することができる（ステップＳ６０５）。

【０１１０】

再び、ステップＳ６０３に戻って、分割の場合は、対応する関数または関数の引数が分割されて増えるため、増えた部分にダミー関数、または、ダミー引数を補填して対応関係をとることにより（ステップＳ６０４）、通常の処理（ステップＳ６０５）が行えるようになる。

【０１１１】

図７は、ＶＡＳ１４０の制御スレッド１４３によりコントロールサービスの特定のインタフェースを隠蔽するためのラッピング処理を行う手順を示すフローチャートである。

【０１１２】

図７において、コントロールサービスからアプリに対して全てメッセージ通知（例えば、イベント通知）が行われるとすると、複合機１００のシステムに大きく影響を与えるコントロールサービスに対して直接アクセスすることが可能となり、複合機のセキュリティ面および障害発生を未然に防止する面からも好ましく

ない。そこで、実施の形態１では、非通知にできるイベントを予め設定することができる。図７に示すように、非通知のイベントを設定する場合は（ステップＳ７０１）、特定のイベントを指定してラッピング処理情報ファイル２０１に対して非通知設定を行う（ステップＳ７０２）。

【０１１３】

そして、図２のコントロールサービス層１５０からアプリに向けてメッセージ通知がなされる場合、ラッピング処理スレッド１４１によりラッピング処理情報ファイル２０１を参照し、非通知設定がなされていればアプリに通知せずに隠蔽する（ステップＳ７０４）。また、上記ステップＳ７０１において非通知設定自体が行われていないか、ステップＳ７０３において非通知設定されていないイベントがある場合は、通常通りアプリにメッセージが通知される（ステップＳ７０５）。ここでは、特定のイベントをアプリに通知するか否かを任意かつ事前に設定できるようにしたため、複合機のセキュリティ面および障害発生を未然に防止することが可能になる。また、実施の形態１では、新規アプリケーションを開発するサードベンダなどの第三者に特定のインタフェースを開示する必要があるため、開示できるインタフェースを自由に選択できるようにすることは非常に重要である。

【０１１４】

図８は、ＶＡＳ１４０の制御スレッド１４３によりアプリ実行の初期段階でバージョン不整合があるか否かを判定するバージョン管理のための手順を示すフローチャートである。

【０１１５】

まず、ＶＡＳ１４０によるバージョンチェック時期としては、アプリが実行される前であれば良く、例えばここではアプリ登録時に行ったが、これに限定されない。そこで、図２のディスパッチャ１４４が起動されたアプリからアプリ登録要求メッセージを受信すると、アプリ登録要求メッセージをそのアプリのプロセスＩＤとともに制御スレッド１４３に受け渡す。制御スレッド１４３は、アプリ登録要求メッセージをプロセスＩＤをディスパッチャ１４４から受信すると、アプリを識別するアプリＩＤを決定して、ＲＡＭ２１０などに格納したアプリ登録

テーブル（図示せず）にアプリＩＤを記録することにより、アプリ登録が行われる。なお、アプリＩＤは、コピーアプリ１１２、プリンタアプリ１１１など既存のアプリケーションについては、予め定められており、各アプリＩＤをＶＡＳ１４０の内部で保持している。また、サードベンダなどが開発した新規アプリ１１７、１１８については、最初の起動時におけるアプリ登録処理の中で決定される。

【０１１６】

このようにして、アプリ登録があった場合（ステップＳ８０１）、アプリごとに持っている一種類のバージョン番号をＶＡＳ１４０に通知する（ステップＳ８０２）。ＶＡＳ１４０では、ＲＡＭ２１０のバージョン管理テーブル２１１（図４参照）とアプリのバージョン番号とを比較し、サポート範囲内か否かを判断する（ステップＳ８０３）。例えば、新規アプリのアプリＩＤが図４中の１０１とすると、そのサポート範囲は、バージョン１．０～１．６であるので、この範囲内のバージョンを持った新規アプリであればサポート範囲内と判定することができる。その判定結果は、当該アプリに対して「ＯＫ」が通知されると（ステップＳ８０４）、正常にアプリ登録され（ステップＳ８０５）、通常の複合機１００における処理が行われる（ステップＳ８０６）。また、ステップＳ８０３でアプリのバージョン番号がバージョン管理テーブル２１１のサポート範囲外と判定されると、当該アプリに対して「ＮＧ」が通知され（ステップＳ８０７）、アプリ登録が終了する（ステップＳ８０８）。

【０１１７】

この図８の例では、アプリのバージョン番号を用いてバージョンを一括管理するため、アプリのバージョン番号をバージョン管理テーブル２１１で単純に番号比較することから、容易に管理することができ、バージョンチェック処理に要する負荷が小さくて済む利点がある。

【０１１８】

図９は、ＶＡＳ１４０の制御スレッド１４３によりアプリが実際に使用している関数を全種類抽出し関数単位でバージョンに不整合があるか否かを判定するバージョン管理の手順を示すフローチャートである。

【０１１９】

図９の場合も、図８と同様にＶＡＳ１４０によるバージョンチェック時期として、アプリが実行される前であれば良く、ここではアプリ登録時を想定している。このようなアプリ登録があった場合（ステップＳ９０１）、アプリがＶＡＳ１４０に対して使用している全種類の関数を抽出し（ステップＳ９０２）、その抽出された関数番号とそのバージョン番号をＶＡＳ１４０に対して使用可能か否か問い合わせる。ＶＡＳ１４０では、ＲＡＭ２１０のバージョン管理テーブル２１１（図５参照）とアプリが使用している全ての関数番号とバージョン番号とを比較することにより、サポート範囲内か否かを判断することができる（ステップＳ９０３）。例えば、ここでは新規アプリが使用している関数番号が「２」、「３」であり、そのバージョン番号が「バージョン１．２」と「バージョン２．３」であって、それぞれの関数番号に対応したサポート範囲もバージョン管理テーブル２１１に格納されているため、関数ごとの判定結果を全てチェックした後、その判定結果を当該アプリに対して通知する。図９の場合、ステップＳ９０３においてサポート範囲内と判定された判定結果は、上記した図８のＡ以降と同様に処理され、ステップＳ９０３においてサポート範囲内でないと判定された場合は、図８のＢ以降と同様に処理されるため、ここでは重複説明を省略する。

【０１２０】

この図９の例では、アプリがＶＡＳ１４０に対して実際に使用している関数番号とそのバージョン番号とを全て抽出し、バージョン管理テーブル２１１に対して関数ごとに比較するため、仮にアプリが実際に使用していない関数のバージョンが変更されたとしても、バージョンチェック結果に影響が出ず、実質的な判定結果が得られることから、より適応範囲を広げることができる。また、個々の関数でサポート範囲外と判定されて使用不可となった場合でも、その原因となった関数が容易に特定できるため、迅速に対応することができる。

【０１２１】

このように、実施の形態１にかかる複合機１００では、仮想アプリケーションサービス１４０のラッピング処理スレッド１４１により、バージョン差を吸収して整合性を保持することができるため、コントロールサービスとそのＡＰＩがバ

ージョンアップしたとしてもその上のアプリをコンパイルし直す必要が無くなる。また、ラッピング処理スレッド１４１は、コントロールサービスからアプリに対するメッセージを取捨選択できるように事前に設定することができるため、複合機１００のシステムに大きく影響を与えるようなインタフェースについては、第三者に対して隠蔽することで、直接コントロールサービスに対してアクセスすることが回避でき、複合機のセキュリティ面および障害発生の未然防止の面から好ましい。また、実施の形態１にかかる複合機１００では、仮想アプリケーションサービス１４０のバージョン管理スレッド１４２により、アプリのバージョン番号を用いてバージョン番号をバージョン管理テーブル２１１を用いて番号比較することにより、容易に管理することができる上、バージョンチェック処理の負荷が小さくて済む。さらに、仮想アプリケーションサービス１４０のバージョン管理スレッド１４２により、アプリがＶＡＳ１４０に対して実際に使用している関数番号とそのバージョン番号とを全て抽出して、バージョン管理テーブル２１１に対して関数ごとに比較を行うため、関数が変更されたことによる影響が少なくなり、実質的に適応範囲が広がり、個々の関数比較によりサポート範囲外と判定された場合でも、原因の関数を容易に特定することができるため、迅速な対応が可能となる。また、実施の形態１にかかる複合機１００では、上記したようにＶＡＳ１４０に加えてダイナミックリンクライブラリ（ＤＬＬ）を採用することも可能であり、その場合はアプリケーションとコントロールサービスあるいはインタフェース間のバージョン差の吸収力が増大し、より柔軟に整合性がとれるようになると共に、インタフェースを隠蔽することができるためインタフェースの秘匿性をより強固に確保することが可能となる。

【０１２２】

（実施の形態２）

実施の形態１にかかる複合機１００は、ＶＡＳ１４０が全アプリケーションに対して１つのみ存在するものであった、この実施の形態２にかかる複合機では、各アプリごとに一つのＶＡＳが起動し、各ＶＡＳは対応するアプリに対してのみバージョン管理およびラッピング処理を行うものである。

【０１２３】

図１０は、実施の形態２にかかる複合機８００の構成を示すブロック図である。図１０に示すように、複合機８００では、複数の仮想アプリケーションサービス（ＶＡＳ）８４１～８４８がアプリケーション１３０の各アプリごとに動作している点が、実施の形態１にかかる複合機１００と異なっている。

【０１２４】

ＶＡＳ８４１～８４８は、プリンタアプリ１１１、コピーアプリ１１２、ファックスアプリ１１３、スキャナアプリ１１４、ネットファイルアプリ１１５、工程検査アプリ１１６、新規アプリ１１７および１１８に対応して、バージョン管理およびラッピング処理を行うようになっている。

【０１２５】

図１１は、実施の形態２にかかる複合機８００のＶＡＳ８４１～８４８の構成と、ＶＡＳ８４１～８４８と各アプリ、コントロールサービス層１５０および汎用ＯＳ１２１との関係を示すブロック図である。なお、図１０では、アプリケーション１３０として、プリンタアプリ１１１、コピーアプリ１１２、新規アプリ１１７、１１８の例を示し、さらにこれら各アプリに対応したＶＡＳ８４１、８４２、８４７および８４８を例として示したが、他のアプリの場合も同様の構成となる。

【０１２６】

また、実施の形態２にかかる複合機８００では、実施の形態１の複合機１００と異なり、図１１に示すように、各ＶＡＳ８４１～８４８と各アプリとの間にはＶＡＳ制御プロセス（デーモン）８０１が動作している。

【０１２７】

このＶＡＳ制御プロセス（デーモン）８０１は、各アプリからアプリ登録要求メッセージを受信してアプリ登録処理を行うとともに、アプリ登録要求を行ったアプリに対応したＶＡＳ８４１～８４８を生成する。また、ＶＡＳ制御プロセス８０１は、ＨＤ２００に格納されたラッピング処理情報ファイル２０１を参照して、アプリ登録要求を行ったアプリについて、バージョン差を吸収して整合性を保つようにしたり、コントロールサービス層１５０からのメッセージを設定した通りに取捨選択することによりインタフェースを隠蔽し、重要なインタフェース

の秘匿性を確保する。

【０１２８】

そして、仮想アプリケーションサービス（ＶＡＳ）８４１～８４８のプロセスには、ディスパッチャ１４４と、ラッピング処理スレッド１４１と、バージョン管理スレッド１４２とが動作している。

【０１２９】

ディスパッチャ１４４は、アプリケーション１３０やコントロールサービスからのメッセージ受信を監視し、受信したメッセージに応じてラッピング処理スレッド１４１、あるいはバージョン管理スレッド１４２に対して処理要求を行うものである。実施の形態２の複合機８００では、ディスパッチャ１４４は、ＶＡＳ制御プロセス８０１から、アプリＩＤ、アプリのプロセスＩＤとともに、バージョン管理要求メッセージまたはラッピング処理要求メッセージを受信するようになっている。ディスパッチャ１４４は、バージョン管理要求メッセージを受信したときには、アプリＩＤ、アプリのプロセスＩＤとともに受信したバージョン管理要求メッセージをバージョン管理スレッド１４２に送信し、ラッピング処理要求メッセージを受信したときには、アプリＩＤ、アプリのプロセスＩＤとともに受信したラッピング処理要求メッセージをラッピング処理スレッド１４１に送信するようになっている。

【０１３０】

バージョン管理スレッド１４２は、ディスパッチャ１４４からのバージョン管理要求メッセージを受信すると、実施の形態１におけるＶＡＳ１４０と同様に、バージョン管理に必要な情報を取得してラッピング処理情報ファイル２０１をハードディスク（ＨＤ）２００に生成する。

【０１３１】

ラッピング処理スレッド１４１は、ディスパッチャ１４４からのラッピング処理要求メッセージを受信すると、実施の形態１におけるＶＡＳ１４０と同様に、ラッピング処理情報ファイル２０１を参照して、ラッピング処理が行われる。

【０１３２】

実施の形態２の複合機８００におけるＶＡＳ８４１～８４８のバージョン管理

スレッド１４２によって実行されるバージョン管理処理、およびラッピング処理スレッド１４１によって実行されるラッピング処理については、実施の形態１の複合機１００におけるＶＡＳ１４０の各スレッドによる処理と同様であるので、重複説明を省略する。

【０１３３】

このように実施の形態２にかかる複合機８００によれば、実施の形態１にかかる複合機１００と同様に、アプリケーションとコントロールサービスおよびインタフェースとの間に生じたバージョン差の整合性を判断し、サポート可能な範囲のバージョン差であればこれを吸収すると共に、アプリケーションとコントロールサービス間の重要なインタフェースを隠蔽して秘匿性を確保しつつ、特定のインタフェースは開示して新規アプリケーションの開発効率を向上させることができる。

【０１３４】

また、実施の形態２にかかる複合機８００では、ＶＡＳ８４１～８４８は起動されるアプリケーション１３０ごとに別個に起動されるので、複数のアプリケーション１３０の起動判断処理を各アプリケーション１３０に対応するＶＡＳ８４１～８４８で並列に実行することができ、アプリケーションの起動判断処理が効率的に行えると共に、上記したバージョン管理やラッピング処理がアプリごとに行えるので、新規アプリの追加や交換時にも効率良く行うことができる。

【０１３５】

なお、実施の形態１および２にかかる複合機１００、８００では、ラッピング処理情報ファイル２０１をハードディスク２００に格納し、バージョン管理テーブル２１１をＲＡＭ２１０に格納して実施したが、かかる格納場所や格納形態は一例であり、他の格納場所や格納形態を採用することも勿論可能である。

【０１３６】

【発明の効果】

以上説明したように、請求項１にかかる発明によれば、コントロールサービスのいずれかのコンポーネントがバージョンアップしたとき、ラッピング処理手段によってバージョン差の生じたアプリケーションからの関数呼び出しに対して整

合性をとるようにしたので、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【０１３７】

また、請求項２にかかる発明によれば、アプリケーションプログラムインタフェースがバージョンアップしたとき、ラッピング処理手段によってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるようにしたので、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【０１３８】

また、請求項３にかかる発明によれば、ラッピング処理手段によってコントロールサービスからアプリケーションに対するメッセージを取捨選択するので、サードベンダなどの第三者に対してはコントロールサービスとの間のインタフェースを隠蔽し、重要なインタフェースの秘匿性を確保して、第三者からコントロールサービスに直接アクセスされるのを回避することにより、画像形成装置におけるセキュリティ面が確保され、障害発生を未然に防止することができる。

【０１３９】

また、請求項４にかかる発明によれば、関数または関数の引数が追加されてバージョンアップした場合、ラッピング処理手段はアプリケーションに対して提供する関数または関数の引数を変更しなければ、追加された機能は使えないが既存の機能はそのまま使えるので、追加機能を無視すればアプリケーションを再コンパイルすることなくバージョン差を吸収することができる。

【０１４０】

また、請求項５にかかる発明によれば、関数または関数の引数が分割されてバージョンアップした場合、対応する関数や関数の引数が増えるので、そのままでは関数呼び出しを行うことができず、ラッピング処理手段によって増加した関数や関数の引数に対応するダミーの関数または関数の引数を補填することによって整合性をとることができ、バージョン差を吸収することができる。

【０１４１】

また、請求項６にかかる発明によれば、特定のメッセージをアプリケーション

に対して通知しないように、ラッピング処理手段に対して予め設定することが可能であるので、重要なインタフェースの秘匿性を確保することができる。

【０１４２】

また、請求項７にかかる発明によれば、ラッピング処理手段は仮想アプリケーションサービスに含まれており、この仮想アプリケーションサービスは、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作するので、バージョン差が生じた場合にバージョン差を吸収して整合性をとることができ、また、コントロールサービスからアプリケーションに対するメッセージを選択することによって、インタフェースを隠蔽して重要なインタフェースの秘匿性を確保しつつ、新規アプリケーションのプログラム開発を効率的に行うことができる。

【０１４３】

また、請求項８にかかる発明によれば、ラッピング処理手段は、仮想アプリケーションサービスのプロセス内部にスレッドとして生成され、実行されるので、仮想アプリケーションサービスのプロセス内部で軽快に動作させることができると共に、プロセス内部に複数個のスレッドを生成した場合であっても、それらを並行処理することができる。

【０１４４】

また、請求項９にかかる発明によれば、アプリケーションの実行時に、共通の関数を利用して他のプログラムと結合させることにより実行形式のプログラムを作成する動的リンク手段を備えているので、アプリケーションとコントロールサービスあるいはインタフェース間のバージョン差の吸収力が増大し、整合性が一層とれるようになると共に、インタフェースを隠蔽して重要なインタフェースの秘匿性をより強固に確保することができる。

【０１４５】

また、請求項１０にかかる発明によれば、アプリケーションとコントロールサービスのいずれかのコンポーネントとの間でバージョン差を生じた場合に、そのバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたので、例えばアプリケーションの登録時などにバー

ジョンの整合性を判定すれば、アプリケーションの実行前にバージョンの適合性の有無が判断できることから、障害の発生を未然に防止することができる。

【０１４６】

また、請求項１１にかかる発明によれば、アプリケーションとアプリケーションプログラムインタフェースとの間でバージョン差が生じた場合に、そのバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたので、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行前にバージョンの適合性の有無が判断できることから、障害の発生を未然に防止することができる。

【０１４７】

また、請求項１２にかかる発明によれば、バージョン管理手段がサポート可能な範囲のバージョンか否かをアプリケーションのバージョン番号に基づいて判定するようにしたので、容易かつ正確に判定することができる。

【０１４８】

また、請求項１３にかかる発明によれば、バージョン管理手段がサポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルを備えていて、アプリケーションのバージョン番号をバージョン管理テーブルと比較して判定するようにしたので、サポート可能なアプリケーションか否かを容易かつ正確に判定することができる。

【０１４９】

また、請求項１４にかかる発明によれば、バージョン管理手段によりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、コントロールサービスとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたので、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理手段によるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【０１５０】

また、請求項１５にかかる発明によれば、バージョン管理手段によりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、アプリケーションプログラムインタフェースとのバージョンとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたので、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理手段によるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【０１５１】

また、請求項１６にかかる発明によれば、バージョン管理手段がサポート可能な範囲のバージョンか否かを判定するにあたって、アプリケーションが使用する全ての関数の番号とバージョン番号に基づいて行われるので、単一な番号比較だけでできることから管理が容易となり、バージョンチェック処理の負荷が小さくなるという利点がある。

【０１５２】

また、請求項１７にかかる発明によれば、バージョン管理手段は、バージョン管理テーブルに登録されているサポート可能な範囲の全ての関数番号とバージョン番号と、アプリケーションが使用する全ての関数番号とバージョン番号とを比較して判定するようにしたので、バージョン差の生じたアプリケーションがサポート可能か否かを容易かつ正確に判定することができる。

【０１５３】

また、請求項１８にかかる発明によれば、バージョン管理手段は、コントロールサービスをサーバとしたクライアントプロセスとして動作し、かつアプリケーションをクライアントとしたサーバプロセスとして動作する仮想アプリケーションサービスに含まれているので、バージョン差が生じた場合にバージョン差を吸収して整合性をとることができ、また、コントロールサービスからアプリケーションに対するメッセージを選択することによって、インタフェースを隠蔽して重要なインタフェースの秘匿性を確保しつつ、新規アプリケーションのプログラム

開発を効率的に行うことができる。

【０１５４】

また、請求項１９にかかる発明によれば、バージョン管理手段は、仮想アプリケーションサービスのプロセス内部にスレッドとして生成され、実行されるので、仮想アプリケーションサービスのプロセス内部で軽快に動作させることができると共に、プロセス内部に複数個のスレッドを生成した場合であっても、それらを並行処理することができる。

【０１５５】

また、請求項２０にかかる発明によれば、コントロールサービスのいずれかのコンポーネントがバージョンアップしたとき、ラッピング処理ステップによってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるので、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【０１５６】

また、請求項２１にかかる発明によれば、アプリケーションプログラムインタフェースがバージョンアップしたとき、ラッピング処理ステップによってバージョン差の生じたアプリケーションからの関数呼び出しに対して整合性をとるようにしたので、バージョン差が吸収され、バージョン不整合による不具合発生を防止することができる。

【０１５７】

また、請求項２２にかかる発明によれば、ラッピング処理ステップによってコントロールサービスからアプリケーションに対するメッセージを取捨選択するので、サードベンダなどの第三者に対してコントロールサービスとの間のインタフェースを隠蔽することで、重要なインタフェースの秘匿性を確保し、第三者からコントロールサービスに直接アクセスされるのを回避して、画像形成装置におけるセキュリティ面を確保することにより、障害の発生が未然に防止できる。

【０１５８】

また、請求項２３にかかる発明によれば、ラッピング処理ステップにおいて、関数または関数の引数が追加されてバージョンアップした場合、追加された関数

または関数の引数に対応する機能を無視して整合性をとるようにしたので、アプリケーションからの関数呼び出しに使われる関数または関数の引数をそのまま使うことができ、アプリケーションを再コンパイルすることなくバージョン差を吸収することができる。

【０１５９】

また、請求項２４にかかる発明によれば、ラッピング処理ステップにおいて、関数または関数の引数が分割されてバージョンアップした場合、分割された関数または関数の引数に応じてダミーの関数または関数の引数を補填して整合性をとるようにしたので、関数の対応関係を維持することが可能となり、アプリケーションを再コンパイルすることなくバージョン差を吸収することができる。

【０１６０】

また、請求項２５にかかる発明によれば、ラッピング処理ステップにおいて、アプリケーションに対して通知しない特定のメッセージが予め設定可能なので、サードベンダなどに公開するインタフェースの基準を任意に決定することができ、重要なインタフェースの秘匿性を確保することができる。

【０１６１】

また、請求項２６にかかる発明によれば、アプリケーションとコントロールサービスのいずれかのコンポーネントとの間でバージョン差が生じた場合に、バージョン管理ステップによりバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたので、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行前にバージョンの適合性の有無が判断でき、障害の発生を未然に防止することができる。

【０１６２】

また、請求項２７にかかる発明によれば、アプリケーションとアプリケーションプログラムインタフェースとの間でバージョン差が生じた場合に、バージョン管理ステップによりバージョン差がサポート可能な範囲か否かを判定し、その判定結果をアプリケーションに通知するようにしたので、例えばアプリケーションの登録時などにバージョンの整合性を判定すれば、アプリケーションの実行前に

バージョンの適合性の有無が判断でき、障害の発生を未然に防止することができる。

【０１６３】

また、請求項２８にかかる発明によれば、バージョン管理ステップにおいて、サポート可能な範囲のバージョンか否かをアプリケーションのバージョン番号に基づいて判定するようにしたので、容易かつ正確に判定することができる。

【０１６４】

また、請求項２９にかかる発明によれば、バージョン管理ステップにおいて、サポート可能な範囲のバージョン番号が登録されたバージョン管理テーブルを用い、登録されるアプリケーションのバージョン番号をバージョン管理テーブルと比較して判定するようにしたので、サポート可能なアプリケーションか否かを容易かつ正確に判定することができる。

【０１６５】

また、請求項３０にかかる発明によれば、バージョン管理ステップによりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、コントロールサービスとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたので、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理ステップによるバージョンチェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【０１６６】

また、請求項３１にかかる発明によれば、バージョン管理ステップによりアプリケーションの使用する関数を全種類抽出し、その抽出した各関数と、アプリケーションプログラムインタフェースとのバージョンとの間にバージョン差が生じた場合に、サポート可能な範囲のバージョン差か否かを判定して、その判定結果をアプリケーションに通知するようにしたので、アプリケーションが使用していない関数がバージョンアップされてもバージョン管理ステップによるバージョン

チェックに影響を与えないことから、サポート可能な範囲を広げることができる。また、サポートが不可能なバージョン差があると判定されても、その原因となる関数を容易に特定することができる。

【０１６７】

また、請求項３２にかかる発明によれば、バージョン管理ステップによりサポート可能な範囲のバージョンか否かを判定するにあたって、アプリケーションが使用する全ての関数の番号とバージョン番号に基づいて行われるので、単一な番号比較だけでできることから管理が容易となり、バージョンチェック処理の負荷が小さくなるという利点がある。

【０１６８】

また、請求項３３にかかる発明によれば、バージョン管理ステップでは、バージョン管理テーブルに登録されているサポート可能な範囲の全ての関数番号とバージョン番号と、アプリケーションが使用する全ての関数番号とバージョン番号とを比較して判定するようにしたので、バージョン差の生じたアプリケーションがサポート可能か否かを容易かつ正確に判定することができる。

【図面の簡単な説明】

【図１】

実施の形態１にかかる複合機の構成を示すブロック図である。

【図２】

実施の形態１にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。

【図３】

ＨＤに格納されるラッピング処理情報ファイルの内容例を示す説明図である。

【図４】

ＲＡＭに格納される一括管理用のバージョン管理テーブルの内容例を示す説明図である。

【図５】

ＲＡＭに格納される関数単位管理用のバージョン管理テーブルの内容例を示す説明図である。

【図６】

ＶＡＳの制御スレッドによりバージョン差を吸収するためのラッピング処理を行う手順を示すフローチャートである。

【図７】

ＶＡＳの制御スレッドによりコントロールサービスの特定のインタフェースを隠蔽するためのラッピング処理を行う手順を示すフローチャートである。

【図８】

ＶＡＳの制御スレッドによりアプリ実行の初期段階でバージョン不整合があるか否かを判定するバージョン管理のための手順を示すフローチャートである。

【図９】

ＶＡＳの制御スレッドによりアプリが実際に使用している関数を全種類抽出し関数単位でバージョンの不整合があるか否かを判定するバージョン管理の手順を示すフローチャートである。

【図１０】

実施の形態２にかかる複合機の構成を示すブロック図である。

【図１１】

実施の形態２にかかる複合機のＶＡＳの構成と、ＶＡＳと各アプリ、コントロールサービス層および汎用ＯＳとの関係を示すブロック図である。

【符号の説明】

１００　複合機
１０１　白黒レーザプリンタ
１０２　カラーレーザプリンタ
１０３　ハードウェアリソース
１１０　ソフトウェア群
１１１　プリンタアプリ
１１２　コピーアプリ
１１３　ファックスアプリ
１１４　スキャナアプリ
１１５　ネットファイルアプリ

１１６　工程検査アプリ

１１７、１１８　新規アプリ

１２０　プラットホーム

１２１　汎用ＯＳ

１２２　ＳＣＳ

１２３　ＳＲＭ

１２４　ＥＣＳ

１２５　ＭＣＳ

１２６　ＯＣＳ

１２７　ＦＣＳ

１２８　ＮＣＳ

１３０　アプリケーション

１４０、８４１～８４８　仮想アプリケーションサービス（ＶＡＳ）

１４１　ラッピング処理スレッド

１４２　バージョン管理スレッド

１４３　制御スレッド

１４４　ディスパッチャ

１５０　コントロールサービス層

２００　ハードディスク（ＨＤ）

２０１　ラッピング処理情報ファイル

２１０　ＲＡＭ

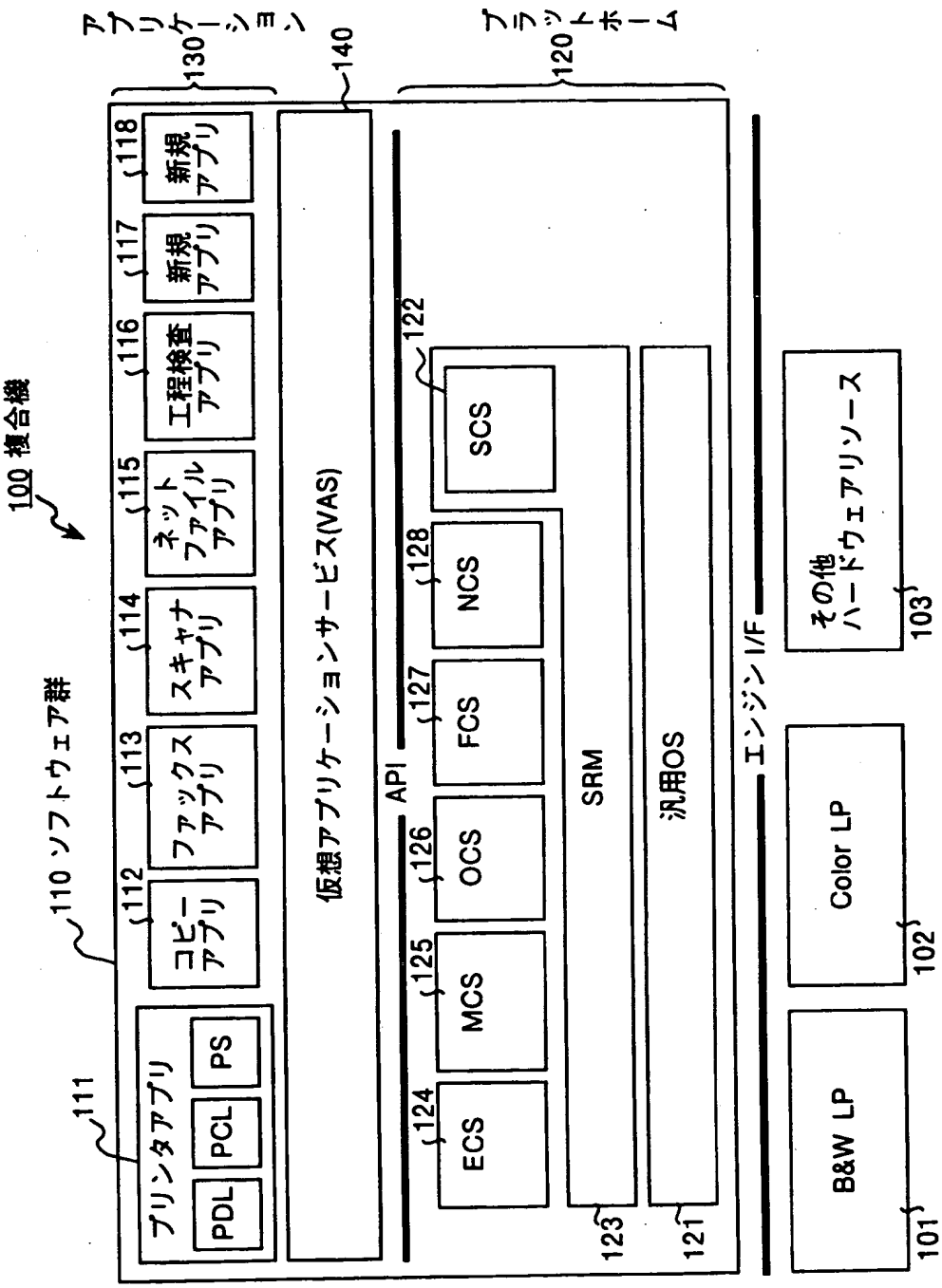
２１１　バージョン管理テーブル

８００　複合機

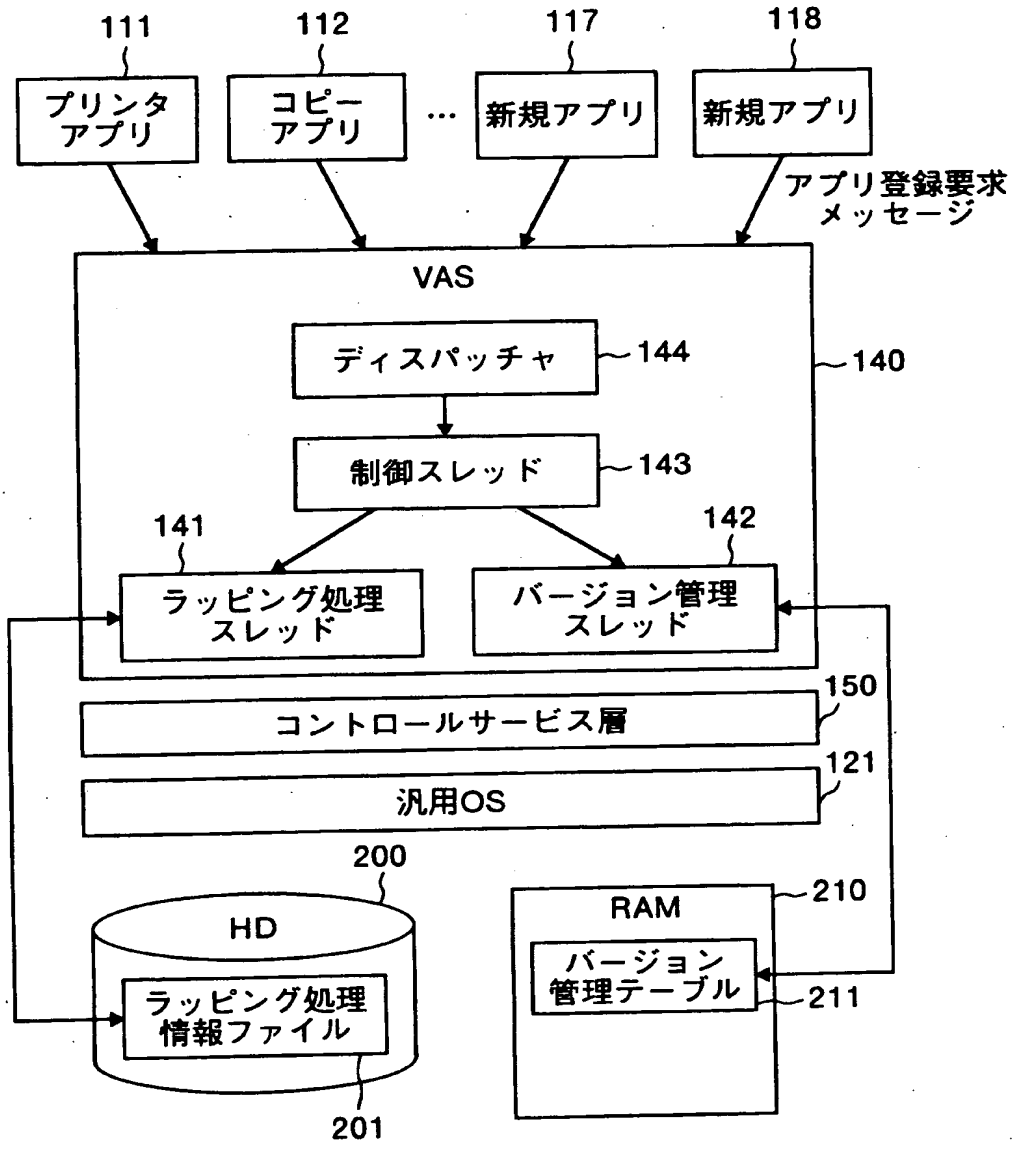
８０１　ＶＡＳ制御プロセス

【書類名】　　図面

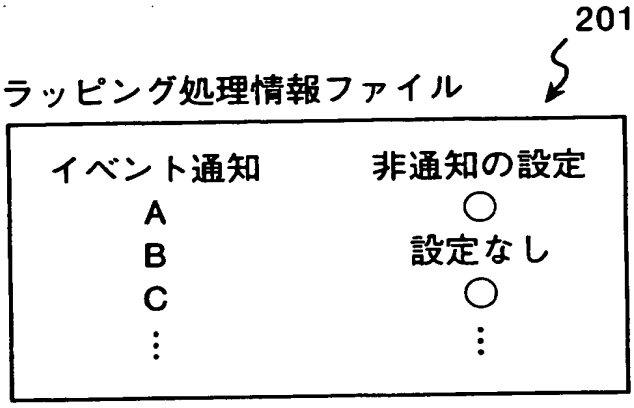
【図１】


100 複合機
110 ソフトウェア群
111
プリンタアプリ
PDL
PCL
PS
112
コピーアプリ
113
ファックスアプリ
114
スキャナアプリ
115
ネットファイルアプリ
116
工程検査アプリ
117
新規アプリ
118
新規アプリ
アプリケーション 130
140
仮想アプリケーションサービス(VAS)
プラットホーム 120
API
122
SCS
128
NCS
127
FCS
126
OCS
125
MCS
124
ECS
SRM
123
汎用OS
121
エンジンI/F
その他ハードウェアリソース
103
Color LP
102
B&W LP
101

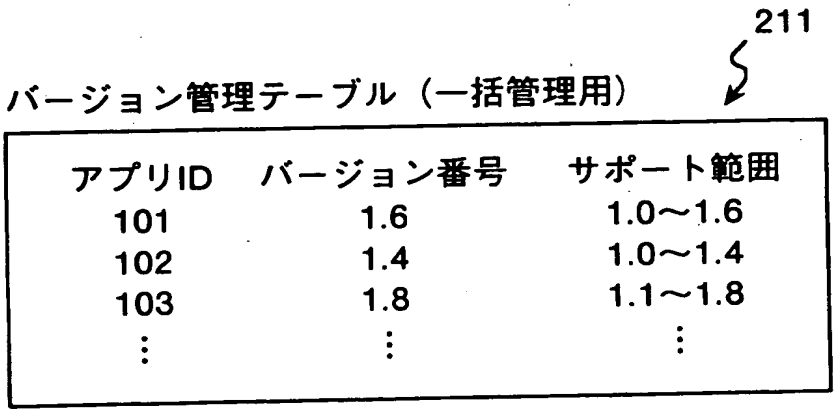
【図２】


111
112
117
118
プリンタアプリ
コピーアプリ
…
新規アプリ
新規アプリ
アプリ登録要求メッセージ
VAS
ディスパッチャ
144
140
制御スレッド
143
141
142
ラッピング処理スレッド
バージョン管理スレッド
150
コントロールサービス層
121
汎用OS
200
HD
ラッピング処理情報ファイル
201
RAM
210
バージョン管理テーブル
211

【図３】


201
ラッピング処理情報ファイル
イベント通知
非通知の設定
A
○
B
設定なし
C
○

【図４】


211
バージョン管理テーブル（一括管理用）
アプリID
バージョン番号
サポート範囲
101
1.6
1.0～1.6
102
1.4
1.0～1.4
103
1.8
1.1～1.8

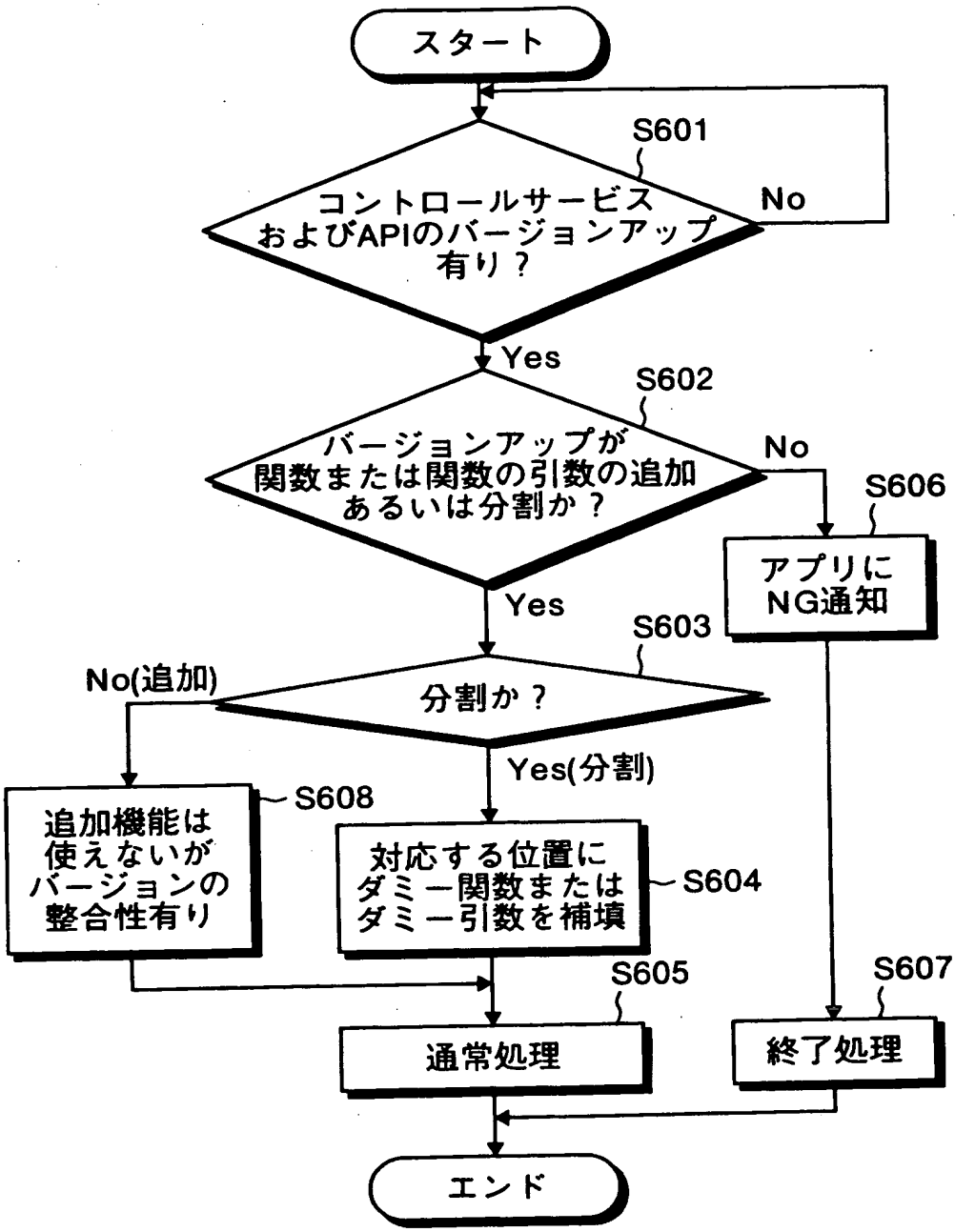
【図５】

211

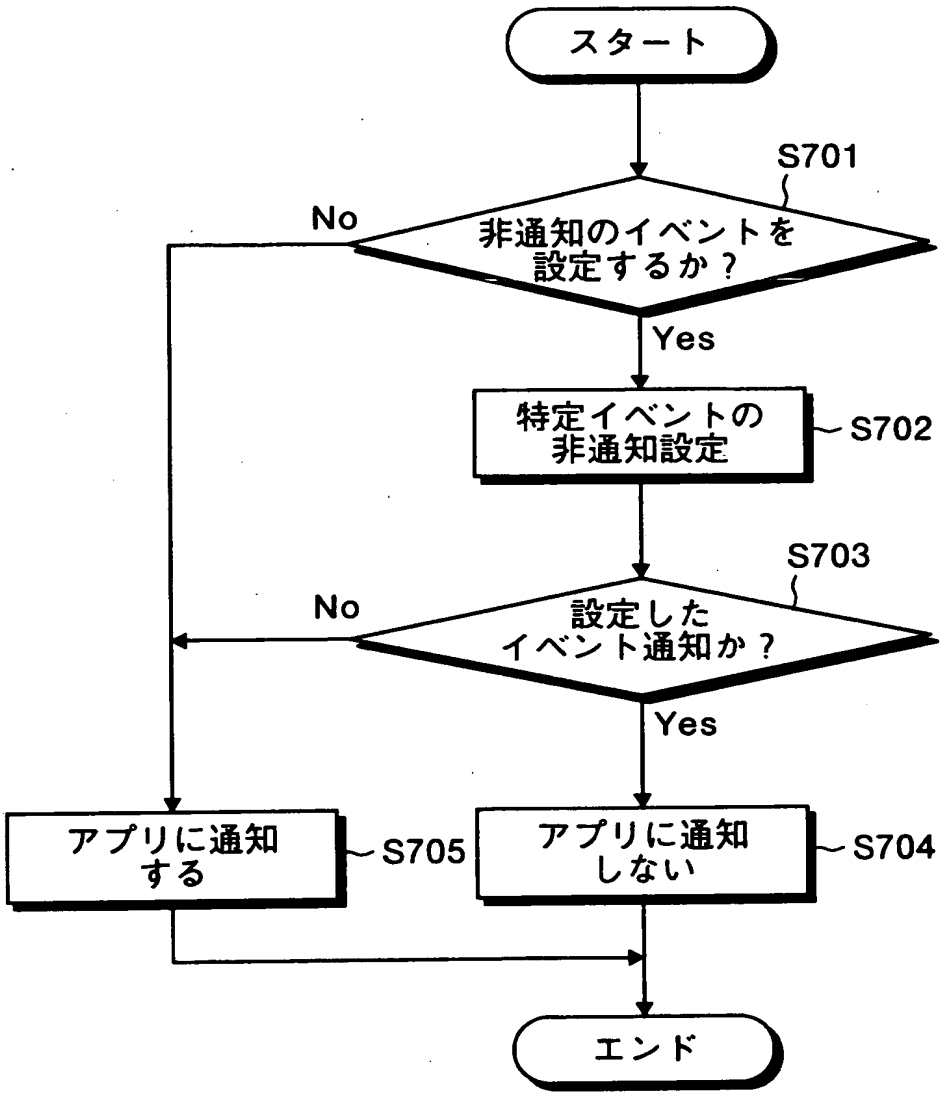
バージョン管理テーブル（関数単位管理用）

関数番号	バージョン番号	サポート範囲	使用の有無
1	1.4	1.0～1.4	×
2	1.2	1.0～1.2	○
3	2.3	2.0～2.3	○
4	1.1	1.0～1.1	×
⋮	⋮	⋮	⋮
n			

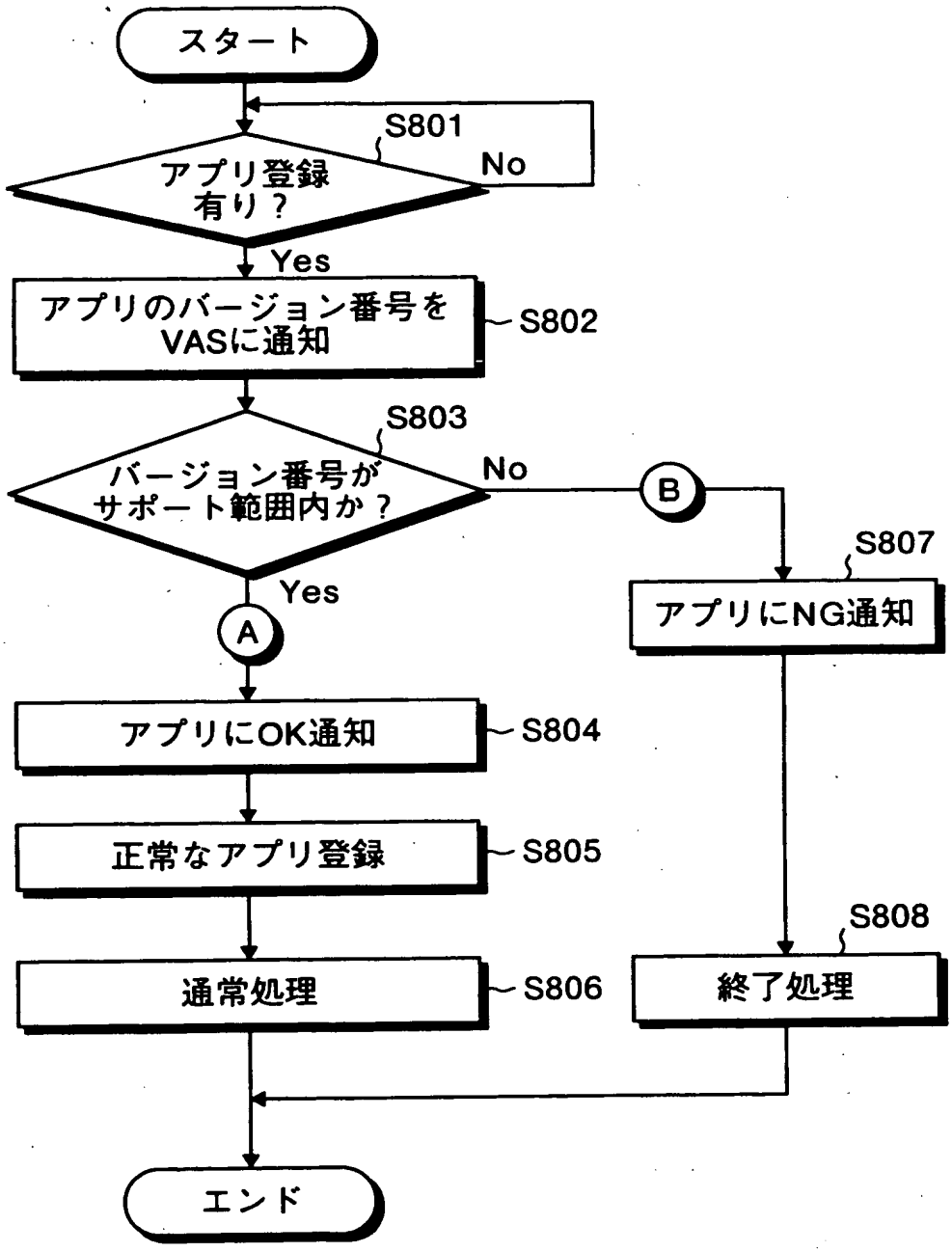
【図６】


スタート
S601
コントロールサービス
およびAPIのバージョンアップ
有り？
No
Yes
S602
バージョンアップが
関数または関数の引数の追加
あるいは分割か？
No
S606
アプリに
NG通知
Yes
S603
No(追加)
分割か？
Yes(分割)
S608
追加機能は
使えないが
バージョンの
整合性有り
対応する位置に
ダミー関数または
ダミー引数を補填
S604
S605
S607
通常処理
終了処理
エンド

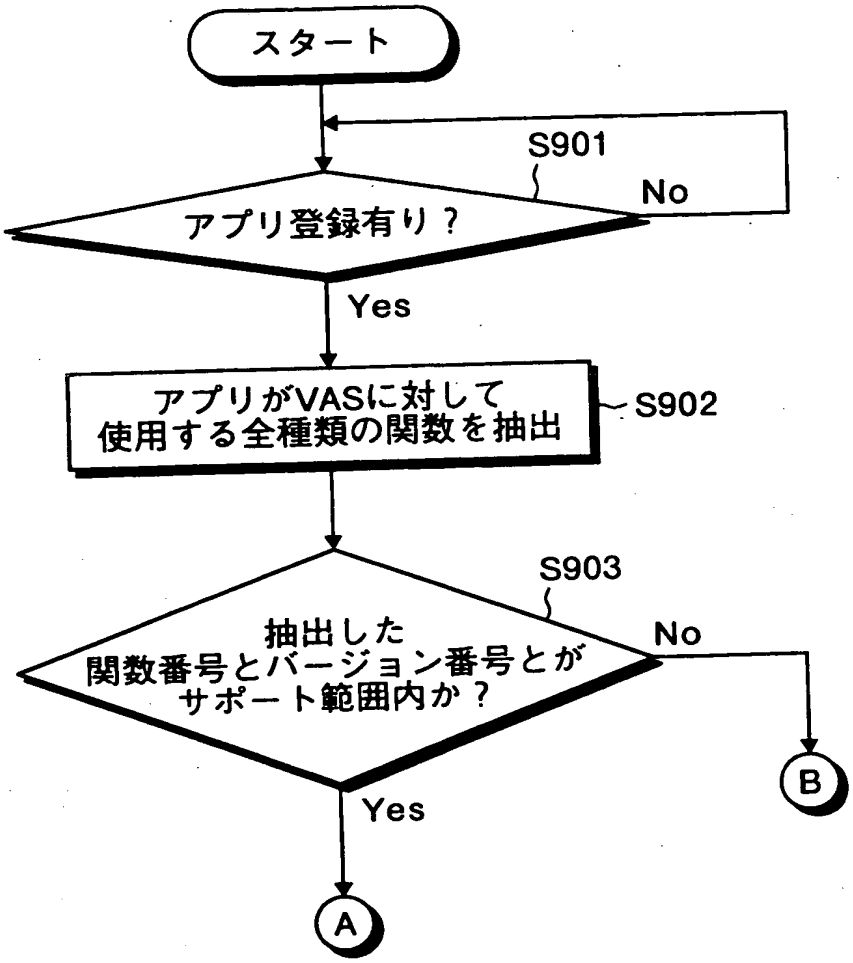
【図７】


スタート
S701
非通知のイベントを
設定するか？
No
Yes
特定イベントの
非通知設定
S702
S703
設定した
イベント通知か？
No
Yes
アプリに通知
する
S705
アプリに通知
しない
S704
エンド

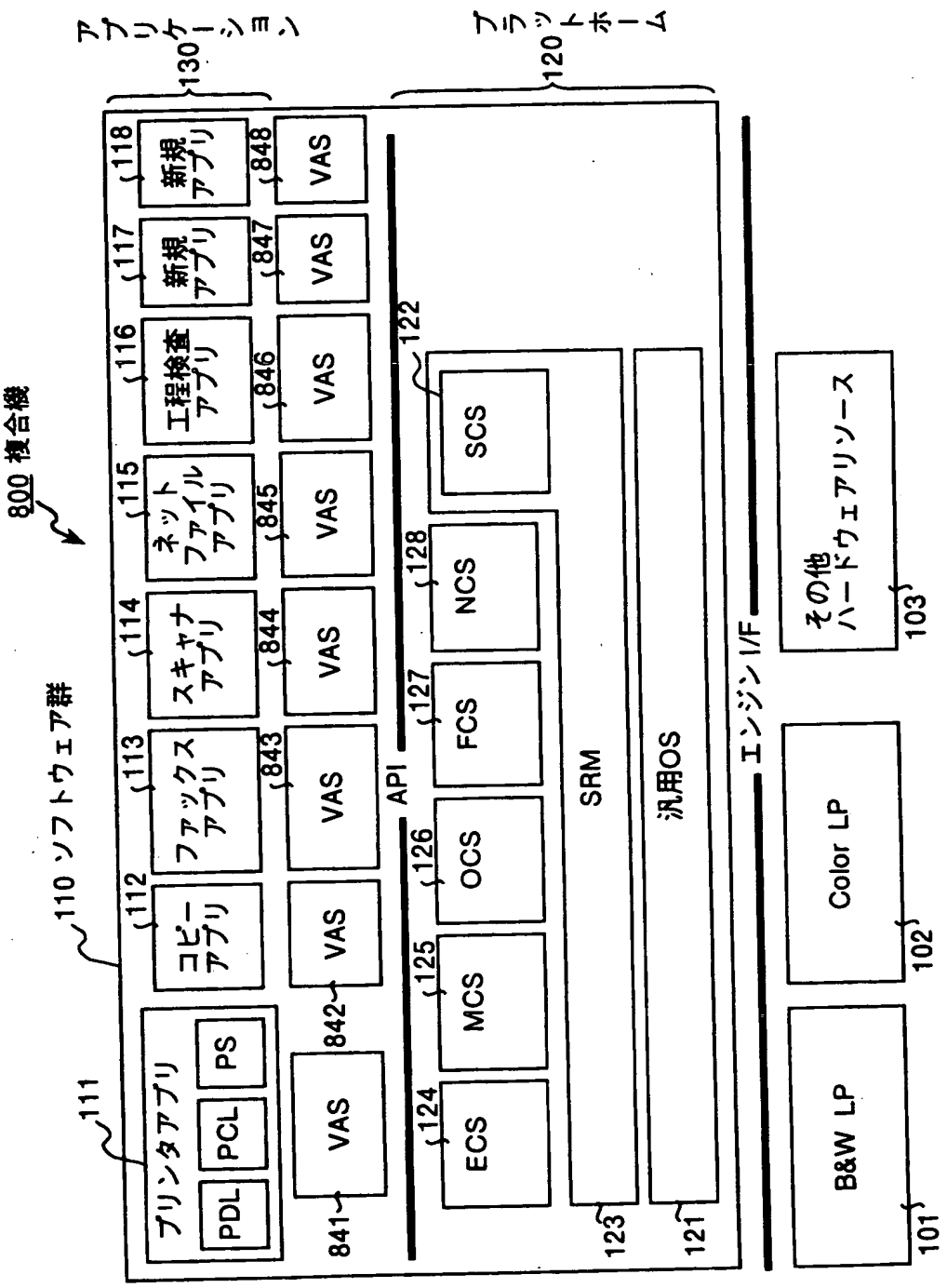
【図８】


スタート
S801
アプリ登録
有り？
No
Yes
アプリのバージョン番号を
VASに通知
S802
S803
バージョン番号が
サポート範囲内か？
No
B
Yes
A
S807
アプリにNG通知
アプリにOK通知
S804
正常なアプリ登録
S805
S808
通常処理
S806
終了処理
エンド

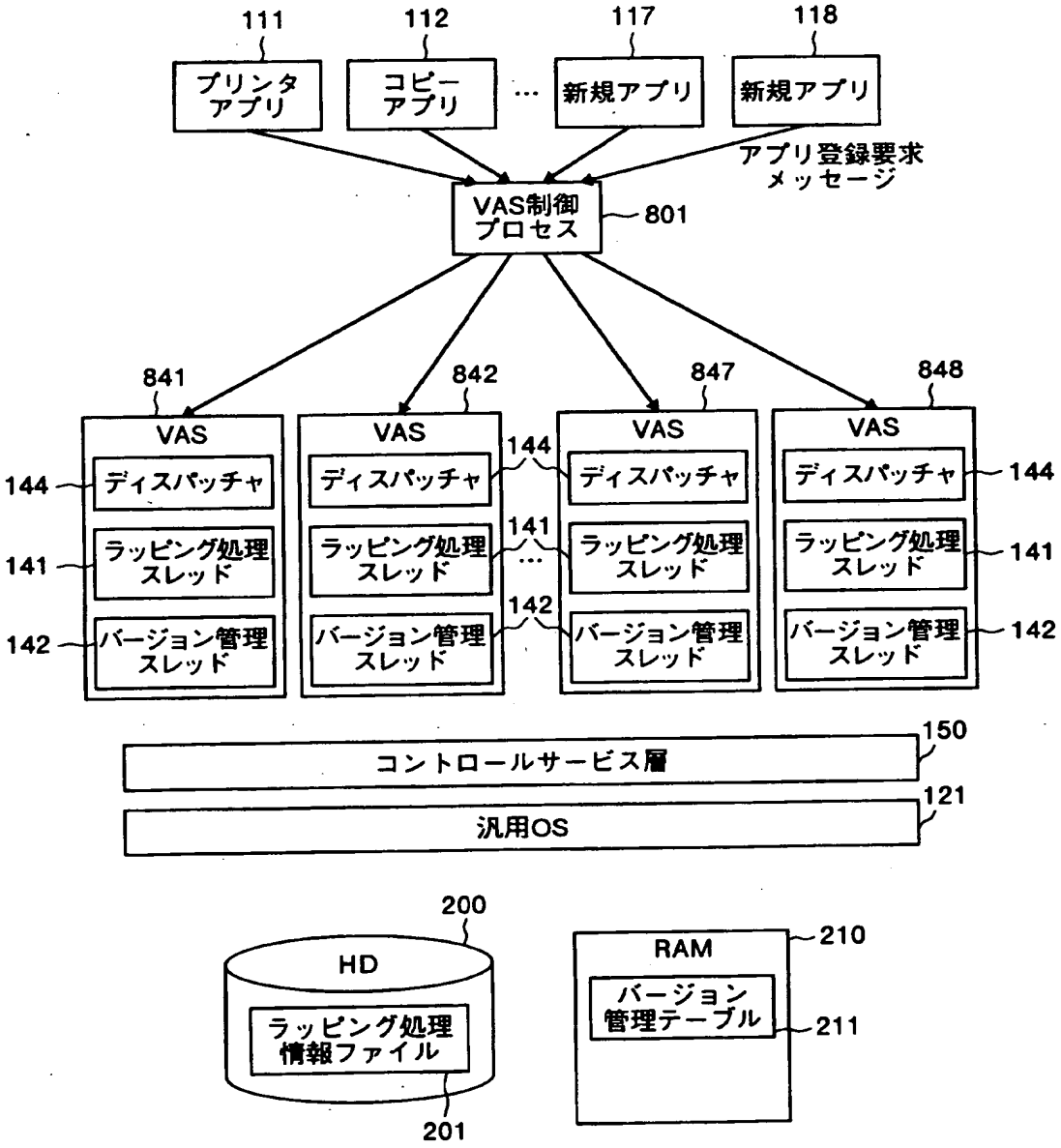
【図９】


スタート
S901
No
アプリ登録有り？
Yes
アプリがVASに対して
使用する全種類の関数を抽出
S902
S903
抽出した
関数番号とバージョン番号とが
サポート範囲内か？
No
Yes
B
A

【図１０】


アプリケーション
130
プラットホーム
120
800 複合機
110 ソフトウェア群
111
プリンタアプリ
PDL
PCL
PS
112
コピーアプリ
113
ファックスアプリ
114
スキャナアプリ
115
ネットファイルアプリ
116
工程検査アプリ
117
新規アプリ
118
新規アプリ
VAS
841
842
843
844
845
846
847
848
API
122
SCS
128
NCS
127
FCS
126
OCS
125
MCS
124
ECS
SRM
123
汎用OS
121
エンジンI/F
その他ハードウェアリソース
103
Color LP
102
B&W LP
101

【図１１】


111
112
117
118
プリンタアプリ
コピーアプリ
新規アプリ
新規アプリ
アプリ登録要求メッセージ
VAS制御プロセス
801
841
842
847
848
VAS
ディスパッチャ
ラッピング処理スレッド
バージョン管理スレッド
144
141
142
コントロールサービス層
150
汎用OS
121
200
HD
ラッピング処理情報ファイル
201
RAM
210
バージョン管理テーブル
211

【書類名】　　要約書

【要約】

【課題】　バージョン差を吸収して、重要なインタフェースの秘匿性を確保しつつ、バージョンチェックを容易かつ確実に行えるようにする。

【解決手段】　ＶＡＳ１４０のラッピング処理スレッド１４１により、バージョン差を吸収して整合性を保持すると共に、コントロールサービス層１５０からアプリに対するメッセージ通知を取捨選択することにより、重要なインタフェースを第三者に対して隠蔽することが可能となる。また、ＶＡＳ１４０のバージョン管理スレッド１４２は、アプリのバージョン番号、あるいは、アプリがＶＡＳ１４０に対して実際に使用している関数番号とそのバージョン番号とをバージョン管理テーブル２１１に対して個々に比較することにより、サポート可能な範囲のバージョン違いか否かを容易かつ確実に判定することができる。

【選択図】　　図２

出　願　人　履　歴　情　報

識別番号　　〔０００００６７４７〕

１．変更年月日　　２００２年　５月１７日
［変更理由］　住所変更
住　所　東京都大田区中馬込１丁目３番６号
氏　名　株式会社リコー